SK Engineering Academy
185102 - Fundamental of Computing and Programming
Question Bank
UNIT – I
PART - A

1. Define computers?

. A computer is a programmable machine or device that performs pre-defined or programmed computations or controls operations that are expressible in numerical or logical terms at high speed and with great accuracy

2. Why computer is known as data processing system?

Any process that uses a computer program will enter data and summarize, analyze or otherwise convert data into usable information. The process may be automated and run on a computer. It involves recording, analyzing, sorting, summarizing, calculating, disseminating and storing data. Thus Computer is known as data processing system.

3. What is Data and Information?

Data - Data is the fact or raw material for the information processing.

Information – The processed data is called information.

4. What are the basic operations of Computer?

It accepts data or instructions by way of input., It stores data., It can process data as required by the user., It gives results in the form of output., It controls all operations inside a computer.

5. Give the applications computer?

Word Processing, Internet, Desktop publishing, Digital video or audio composition, Mathematical Calculations, Robotics, Weather analysis

6. What are the characteristics of computers? (JAN 2011)

Speed, Accuracy, Automation, Endurance, Versatility, Storage, Cost Reduction.

7. How will you classify computer systems? (JAN 2011)

Based on physical size, performance and application areas, we can generally divide computers into four major categories:

Micro computer, Mini Computer, Mainframe computer and Super Computer.

8. Specify the Electronic components used for different computer generations. Generations Electronic Components

I Generation Vacuum tubes, II Generation Transistors, III Generation Integrated Circuits, IV Generation Microprocessors, V Generation Artificial Intelligence

9. Define Software and Hardware? (JAN 2011)

Computer instructions or data, anything that can be stored electronically is software. Computer hardware - (computer science) the mechanical, magnetic, electronic, and electrical components making up a computer system hardware.

10. What are the languages used in computer generations.

Generations Languages used

I Generation Machine Language, II Generation Assemble Language, Mnemonics, III Generation High Level Language, BASIC, PASCAL, COBOL, FORTRON, IV generation 4GL, V Generation Artificial Intelligence.

11. Expand ENIVAC, ABC, EDVAC, EDSAC and UNIVAC.

ENIAC – Electronic Numerical Integrator and Calculator, **ABC –** Atanasoff and Berry Computer, **EDVAC –** Electronic Discrete Variable Automatic Calculator, **EDSAC –** Electronic Delay Storage Automatic Calculator, **UNIVAC –** UNIversal Automatic Computer.

12. Who is the father of computer? Why?

Charles Babbage is the father of computer, because the parts and working principle of the Analytical Engine, which is invented by Charles Babbage is similar to today's computer.

13. Expand COBOL, BASIC, FORTRON and IBM.

COBOL – Common Business Oriented Language, **BASIC -** Beginner's All Purpose Symbolic Instruction Code, **FORTRON –** FORmula TRANslation, **IBM –** International Business Machine.

14. Expand IC, SSI, MSI, LSI, and VLSI.

IC – Integrated Circuit, **SSI -** Small Scale Integration, **MSI -** Medium Scale Integration.
LSI - Large Scale Integration, **VLSI -** Very Large Scale Integration.

15. What are the components of the computer systems?

Basic components of the computer system are Input Unit, Central Processing Unit, Secondary Storage Unit and Output Unit.

16. What are the functions in the input unit?

An input device is a device that is used to input data or information into a computer. Some examples of input devices include:

Keyboards, Computer mice, Light Pen, Digitizer, Touchpad, Trackball, Image scanner, Webcam, Video capture / tuner cards, Microphones, MIDI instruments.

17. What are the functions in the output unit?

In computers, a unit which delivers information from the computer to an external device or from internal storage to external storage.

Speakers, Printer, Headphone, Monitor (or) Visual Display Unit (VDU), Plotter.

18. What is an ALU?

Arithmetic logic unit, the part of a computer that performs all arithmetic computations, such as addition and multiplication, and all logical operations such s comparison operations. The ALU is one component of the CPU (central processing unit).

19. Define Clients and Servers.

A **client** is generally a single-user PC or workstation that provides a highly user-friendly interface to the end user. It runs client processes, which send service requests to the server.

A **server** is generally a relatively large computer that manages a shared resource and provides a set of shared user services to the clients. It runs the server process, which services client requests for use of the resource managed by the server. The network may be single LAN or WAN or an internet of networks.

20. What is a CPU?

The CPU (central processing unit) is the part of a computer controls the interpretation and execution of instructions. Generally, the CPU is a single microchip.The CPU, clock and main memory make up a computer. A complete computer system requires the addition of control units, input, output and storage devices and an operating system.

21. What is meant by generation in computer terminology?

Generation is the period of years in which the computers are enhanced as previous.

22. Define personal computers?

A small, relatively inexpensive computer designed for an individual user. In price, personal computers range anywhere from a few hundred dollars to thousands of dollars. All are based on the microprocessor technology that enables manufacturers to put an entire CPU on one chip.

Example:

- Businesses use personal computers for word processing, accounting, desktop publishing, and for running spreadsheet and database management applications.
- At home, the most popular use for personal computers is for playing games.

23. Define Mainframe computer? Give the name of any one mainframe computer. (JAN 2012)

Mainframes are computers used mainly by large organizations for critical applications, typically bulk data processing such as census, industry and consumer statistics, enterprise resource planning, and financial processing.. Mainframe is an industry term for a large computer. And because of the sheer development costs, mainframes are typically manufactured by large companies such as IBM, Amdahl, Hitachi.

Ex. ENIAC, UNIVAC

24. Define Mini computers?

A mini computer is a multi-user or time-sharing system. It is used for medium scale data processing such as Bank account processing, Payroll processing etc., Mini computer process greater storage capacity and larger memories as compared to micro computer.

25. Define super computer?

The fastest type of computer. Supercomputers are very expensive and are employed for

specialized applications that require immense amounts of mathematical calculations.For example, weather forecasting requires a supercomputer.

26. What is a volatile and non-volatile memory?

Volatile memory: also known as volatile storage is computer memory that requires power to maintain the stored information, unlike non-volatile memory which does not require a maintained power supply. It has been less popularly known as temporary memory.

Non-volatile memory: nonvolatile memory, NVM or non-volatile storage, is computer memory that can retain the stored information even when not powered.

27. Define (1) Nibble (2) Bit (3) Byte?

(1) In computers and digital technology, a nibble is four binary digits or half of an eight-bit byte. A nibble can be conveniently represented by one hexadecimal digit.

(2) A bit (short for binary digit) is the smallest unit of data in a computer. A bit has a Single binary value, either 0 or 1. Although computers usually provide instructions that can test and manipulate bits, they generally are designed to store data and execute instructions in bit multiples called bytes.

(3) In most computer systems, a byte is a unit of data that is eight binary digits long. A byte is the unit most computers use to represent a character such as a letter, number, or typographic symbol (for example, "g", "5", or "?"). A byte can also hold a string of bits that need to be used in some larger unit for application purposes.

28. Write the binary and octal equivalent of hexadecimal number 7BD?

Binary Equivalent of 7BD = $(0111\ 1011\ 1101)_2$

Octal Equivalent of 7BD = $(011\ 110\ 111\ 101) = (3675)_8$

29. Find the octal equivalent of the number. $(10111001100.1101011)_2$

Answer: 2714.654

PART – B

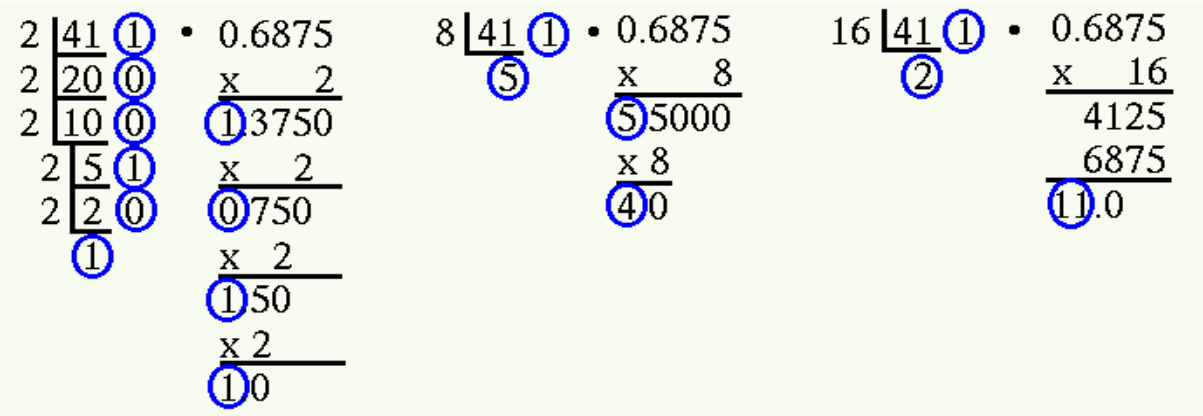
1. Define computer. Explain the characteristics briefly?

A computer is an electronic device that operates under the control of a set of instructions that is stored in its memory unit. A computer accepts data from an input device and processes it into useful information which it displays on its output device. Actually, a computer is a collection of hardware and software components that help you accomplish many different tasks. Hardware consists of the computer itself, and any equipment connected to it. Software is the set of instructions that the computer follows in performing a task. We will explore hardware and software more in depth in the next module. A typical computer system is shown in the figure below.

1. Speed
2. Storage capacity
3. Accuracy
4. Reliability
5. Versatility
6. Diligence/endurance

2. With suitable examples, explain about Number systems. (JAN 2012)

DECIMAL TO BINARY,OCTAL,HEXADECIMAL:



$41.6875_{10} = 101001.1011_2$

$41.6875_{10} = 51.54_8, \quad 5 \times 8^1 + 1 \times 8^0 + 5 \times 8^{-1} + 4 \times 8^{-2} = 41.6875_{10}$

$41.6875_{10} = 29.A_{16} \quad 2 \times 16^1 + 9 \times 16^0 + 11 \times 16^{-1} = 41.6875_{10}$

BINARY TO DECIMAL,OCTAL,HEXADECIMAL:

$110011_2 = 1\times2^5 + 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = 51_{10}$

$$1101011.1011_2 = \underline{001}\ \underline{101}\ \underline{011}.\underline{101}\ \underline{100}_2 = 153.54_8$$

$$1001011.1101 = \underline{0100}\ \underline{1011}.\underline{1101}_2 = 4B.D_{16}$$

OCTAL TO DECIMAL,BINARY,HEXADECIMAL:

$51.54_8 = 5\times8^1 + 1\times8^0 + 5\times8^{-1} + 4\times8^{-2} = 41.6875_{10}$

$27.35_8 = \underline{010}\ \underline{111}.\underline{011}\ \underline{101}_2 = 10111.011101_2$

$27.35_8 = 0001\ \ 0111\ .\ 0111\ \ 0100\ =\ \ 17.74_{16}$

HEXADECIMAL TO DECIMAL,BINARY,OCTAL:

$29.A_{16} = 2\times16^1 + 9\times16^0 + 11\times16^{-1} = 41.6875_{10}$

$27.35_{16} = \underline{0010}\ \underline{0111}.\underline{0011}\ \underline{0101}_2$

$4B.D_{16} = 0\ 100\ 1011\ .\ 1101\ =\ 001\ \ 001\ \ 011\ .\ 110\ \ 100\ =\ 113.64_8$

3. Describe evolution of computer? (JAN 2012)

The term Computer, originally meant a person capable of performing numerical calculations with the help of a mechanical computing device. The evolution of computers started way back in the late 1930s. Binary arithmetic is at the core of the computers of all times. History of computers dates back to the invention of a mechanical adding machine in 1642. ABACUS, an early computing tool, invention of logarithm by John Napier and the invention of slide rules by William Oughtred were significant events in the evolution of computers from these early computing devices. In the evolution of computers their first generation was characterized by the use of vacuum tubes. These computers were expensive and bulky. They used machine language for computing and could solve just one problem at a time. They did not support multitasking.

- It was in 1937 that John V. Atanasoff devised the first digital electronic computer. Atanasoff and Clifford Berry came up with the ABC prototype in the November of 1939. Its computations were based on a vacuum tube and it used regenerative capacitor memory.
- Konrad Zuse's electromechanical 'Z Machines', especially the Z3 of 1941 was a notable achievement in the evolution of computers. It was the first machine to include binary and floating-point arithmetic and a considerable amount of programmability. In 1998, since it was proved to be Turing complete, it is regarded as world's first operational computer.
- In 1943, the Colossus was secretly designed at Bletchley Park, Britain to decode German messages. The Harvard Mark I of 1944 was a large-scale electromechanical computer with less programmability. It was another step forward in the evolution of computers.
- The U.S. Army's Ballistics Research Laboratory came up with the Electronic Numerical Integrator And Computer (ENIAC) in 1946. It came to be known as the first general purpose electronic computer. However it was required to be rewired to change it's programming thus making its architecture inflexible. Developers of ENIAC realized the flaws in the architecture and developed a better architecture. It was known as the stored program architecture or von Neumann Architecture. It got its name after John von Neumann, who for the first time described the architecture in 1945. All the projects of developing computers taken up thereafter have been using the von Neumann Architecture. All the computers use a 'stored program architecture', which is now a part of the definition of the word 'computer'.The U.S. National Bureau of Standards came up with Standards Electronic/Eastern Automatic Computer (SEAC) in 1950. Diodes handled all the logic making it the first computer to base its logic on solid devices. IBM announced the IBM 702 Electronic Data Processing Machine in 1953. It was developed for business use and could address scientific and engineering applications. Till the 1950s all computers that were used were vacuum tube based.

In the 1960s, transistor based computers replaced vacuum tubes. Transistors made computers smaller and cheaper. They made computers energy efficient. But transistors were responsible for the emission of large amounts of heat from the computer. Due to this computers were subject to damage. The use of transistors marked the second generation

of computers. Computers belonging to this generation used punched cards for input. They used assembly language.

- Stanford Research Institute brought about ERMA, Electronic Recording Machine Accounting Project, which dealt with automation of the process of bookkeeping in banking.
- In 1959, General Electric Corporation delivered its ERMA computing system to the Bank of America in California.

The use of Integrated circuits ushered in the third generation of computers. Small transistors placed on silicon chips, called semi conductors. This increased the speed and efficiency of computers. Operating systems were the human interface to computing operations and keyboards and monitors became the input-output devices.

- In 1968, DEC launched the first mini computer called the PDP-8.
- In 1969, the development of ARPANET began with the financial backing of the Department Of Defense.

Thousands of integrated circuits placed onto a silicon chip made up a microprocessor. Introduction of microprocessors was the hallmark of fourth generation computers.

- Intel produced large-scale integration circuits in 1971. During the same year, Micro Computer came up with microprocessor and Ted Hoff, working for Intel introduced 4-bit 4004.
- In 1972, Intel introduced the 8080 microprocessors.
- In 1974, Xerox came up with Alto workstation at PARC. It consisted of a monitor, a graphical interface, a mouse, and an Ethernet card for networking.
- Apple Computer brought about the Macintosh personal computer January 24 1984.

4. Explain the fundamental units of a computer with a block diagram? (JAN 2012) (JAN 2011)

A computer as shown below performs basically five major operations or functions irrespective of their size and make. These are 1) it accepts data or instructions by way of input, 2) it stores data, 3) it can process data as required by the user, 4) it gives results in the form of output, and 5) it controls all operations inside a computer. We discuss below each of these operations.

1. Input: This is the process of entering data and programs in to the computer system. You should know that computer is an electronic machine like any other machine which takes as inputs raw data and performs some processing giving out processed data. Therefore, the input unit takes data from us to the computer in an organized manner for processing.

2. Storage: The process of saving data and instructions permanently is known as storage. Data has to be fed into the system before the actual processing starts. It is because the processing speed of Central Processing Unit (CPU) is so fast that the data has to be provided to CPU with the same speed. Therefore the data is first stored in the storage unit for faster access and processing. This storage unit or the primary storage of the computer system is designed to do the above functionality. It provides space for storing data and instructions.

The storage unit performs the following major functions:

- All data and instructions are stored here before and after processing.
- Intermediate results of processing are also stored here.

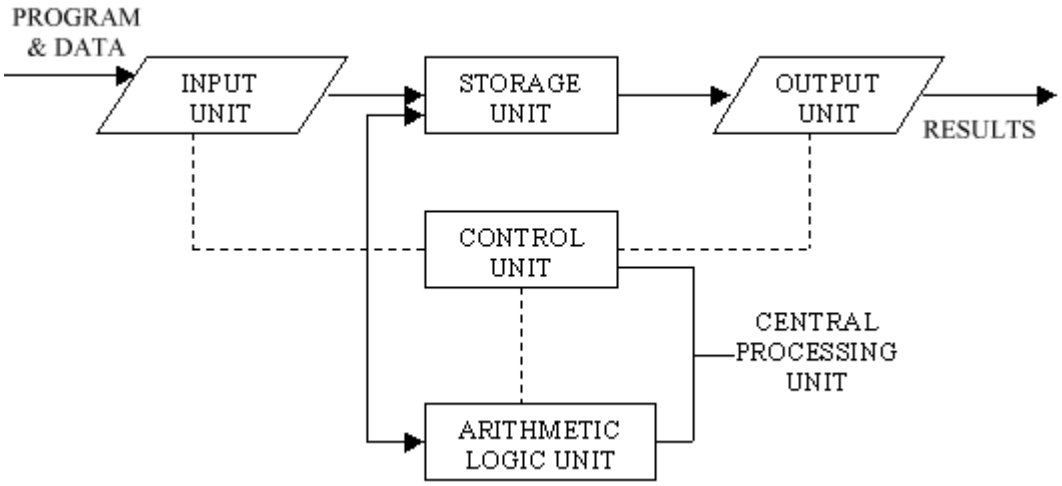


Fig : Basic computer Operations

3. Processing: The task of performing operations like arithmetic and logical operations is called processing. The Central Processing Unit (CPU) takes data and instructions from

the storage unit and makes all sorts of calculations based on the instructions given and the type of data provided. It is then sent back to the storage unit.

4. Output: This is the process of producing results from the data for getting useful information. Similarly the output produced by the computer after processing must also be kept somewhere inside the computer before being given to you in human readable form. Again the output is also stored inside the computer for further processing.

5. Control: The manner how instructions are executed and the above operations are performed. Controlling of all operations like input, processing and output are performed by control unit. It takes care of step by step processing of all operations in side the computer.

FUNCTIONAL UNITS

In order to carry out the operations mentioned in the previous section the computer allocates the task between its various functional units. The computer system is divided into three separate units for its operation. They are 1) arithmetic logical unit, 2) control unit, and 3) central processing unit.

Arithmetic Logical Unit (ALU)

After you enter data through the input device it is stored in the primary storage unit. The actual processing of the data and instruction are performed by Arithmetic Logical Unit. The major operations performed by the ALU are addition, subtraction, multiplication, division, logic and comparison. Data is transferred to ALU from storage unit when required. After processing the output is returned back to storage unit for further processing or getting stored.

Control Unit (CU)

The next component of computer is the Control Unit, which acts like the supervisor seeing that things are done in proper fashion. The control unit determines the sequence in which computer programs and instructions are executed. Things like processing of programs stored in the main memory, interpretation of the instructions and issuing of signals for other units of the computer to execute them. It also acts as a switch board operator when several users access the computer simultaneously. Thereby it coordinates

the activities of computer's peripheral equipment as they perform the input and output. Therefore it is the manager of all operations mentioned in the previous section.

Central Processing Unit (CPU)

The ALU and the CU of a computer system are jointly known as the central processing unit. You may call CPU as the brain of any computer system. It is just like brain that takes all major decisions, makes all sorts of calculations and directs different parts of the computer functions by activating and controlling the operations

Personal Computer Configuration

Now let us identify the physical components that make the computer work. These are

1. Central Processing Unit (CPU)
2. Computer Memory (RAM and ROM)
3. Data bus
4. Ports
5. Motherboard
6. Hard disk
7. Output Devices
8. Input Devices

MEMORY SYSTEM IN A COMPUTER

There are two kinds of computer memory: *primary* and *secondary*. Primary memory is accessible directly by the processing unit. RAM is an example of primary memory. As soon as the computer is switched off the contents of the primary memory is lost. You can store and retrieve data much faster with primary memory compared to secondary memory. Secondary memory such as floppy disks, magnetic disk, etc., is located outside the computer. Primary memory is more expensive than secondary memory. Because of this the size of primary memory is less than that of secondary memory. We will discuss about secondary memory later on.

Computer memory is used to store two things: i) instructions to execute a program and ii) data. When the computer is doing any job, the data that have to be processed are stored in the primary memory. This data may come from an input device like keyboard or from a secondary storage device like a floppy disk.

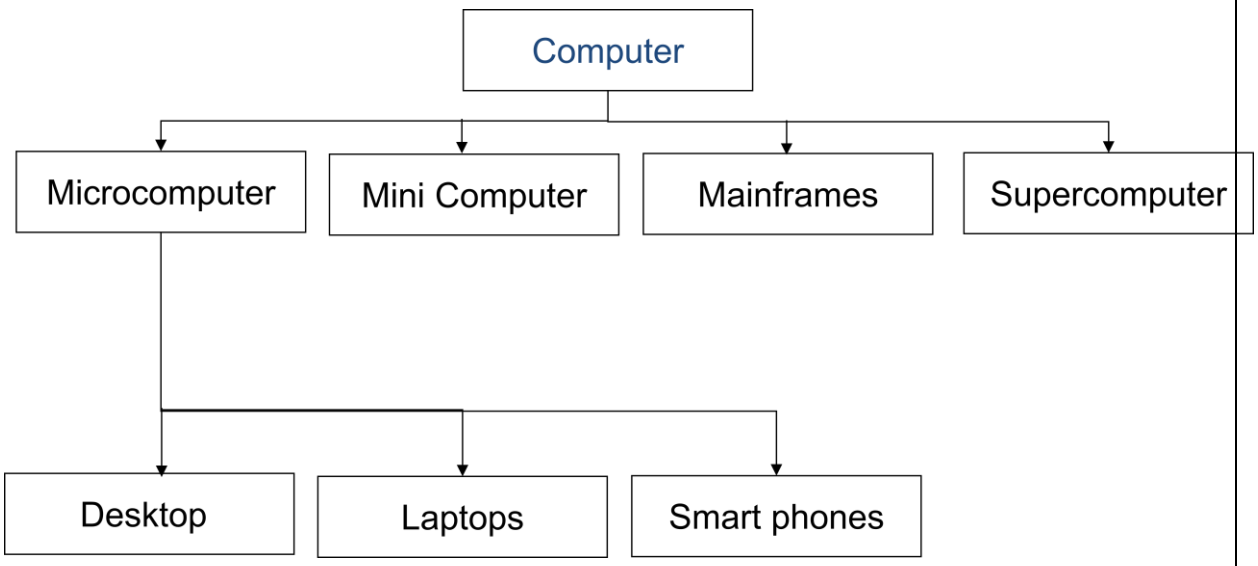
But inside the computer, the steps followed are quite different from what we see on the monitor or screen. In computer's memory both programs and data are stored in the binary form. You have already been introduced with decimal number system, that is the numbers 1 to 9 and 0. The binary system has only two values 0 and 1. These are called *bits*. As human beings we all understand decimal system but the computer can only understand binary system. It is because a large number of integrated circuits inside the computer can be considered as switches, which can be made ON, or OFF. If a switch is ON it is considered 1 and if it is OFF it is 0. A number of switches in different states will give you a message like this: 110101....10. So the computer takes input in the form of 0 and 1 and gives output in the form 0 and 1 only. Is it not absurd if the computer gives outputs as 0's & 1's only? But you do not have to worry about.

Every number in binary system can be converted to decimal system and vice versa; for example, 1010 meaning decimal 10. Therefore it is the computer that takes information or data in decimal form from you, convert it in to binary form, process it producing output in binary form and again convert the output to decimal form.

The primary memory as you know in the computer is in the form of IC's (Integrated Circuits). These circuits are called Random Access Memory (RAM). Each of RAM's locations stores one *byte* of information. (One *byte* is equal to *8 bits*). A bit is an acronym for *binary digit*, which stands for one binary piece of information. This can be either 0 or 1. You will know more about RAM later. The Primary or internal storage section is made up of several small storage locations (ICs) called cells. Each of these cells can store a fixed number of bits called *word length***.**

Each cell has a unique number assigned to it called the address of the cell and it is used to identify the cells. The address starts at 0 and goes up to (N-1). You should know that the memory is like a large cabinet containing as many drawers as there are addresses on memory. Each drawer contains a word and the address is written on outside of the drawer.

5.Explain the classification of computers?



Computers can be classified many different ways -- by size, by function, and/or by processing capacity. We will study the classification of computers by size. The size of a computer often determines its function and processing capacity. The size of computers varies widely from tiny to huge and is usually dictated by computing requirements. For example, it is clear that the IRS will have different requirements than those of a college student.

The largest computers are ***supercomputers***. They are the most powerful, the most expensive, and the fastest. They are capable of processing trillions of instructions per second. Examples of users of these computers are governmental agencies, such as the IRS, the National Weather Service, and the National Defense Agency. Also, they are used in the making of movies, space exploration, and the design of many other machines. The Cray supercomputer is nicknamed "Bubbles", because of its bubbling coolant liquids. Cray supercomputers, the first of which was invented by Seymour Cray, now maintain 75 percent of the supercomputer market. Supercomputers are used for tasks that require mammoth data manipulation.

Large computers are called ***mainframes***. Mainframe computers process data at very high rates of speed, measured in the millions of instructions per second. They are very expensive, costing millions of dollars in some cases. Mainframes are designed for multiple users and process vast amounts of data quickly. Banks, insurance companies, manufacturers, mail-order companies, and airlines are typical users. Mainframes are often

'servers'-- computers that control the networks of computers for large companies like catalog merchandiser L. L. Bean.

Microcomputers can be divided into two groups -- personal computers and workstations. Workstations are specialized computers that approach the speed of mainframes. Often microcomputers are connected to networks of other computers. The price of a microcomputer varies greatly from less than $1000 to several thousand dollars, depending on the capacity and features of the computer. Microcomputers make up the vast majority of computers.

Another classification of computer is the ***notebook*** computer. A notebook computer can fit into a briefcase and weigh fewer than two pounds, yet it can compete with the microcomputer. A larger, heavier version is called a ***laptop*** computer. Notebooks generally cost more than microcomputers but can run most of the microcomputer software and are more versatile. Like other computers, notebook computers are getting faster, lighter, and more functional.

The smallest computer is the handheld computer called a ***personal digital assistant*** or a ***PDA***. PDAs are used to track appointments and shipments as well as names and addresses. PDAs are called pen-based computers because they utilize a pen-like stylus that accepts hand-written input directly on a touch-sensitive screen. You have probably noticed delivery employees using these.

Speed, Reliability, Storage Capacity and Productivity

Computers of all sizes have common characteristics -- speed, reliability, storage capacity, and productivity. Computers are the foundation of business, travel, and leisure life today. Computers provide the processing speed required by all facets of society. The quick service we expect at the bank, at the grocery store, on the stock exchange, and on the Internet are dependent on the speed of computers. Computers are extremely reliable as well. Most errors are caused by humans, not computers. Computers are capable of storing enormous amounts of data that must be located and retrieved very quickly. The capability to store and retrieve volumes of data is at the core of the Information Age

6.Describe briefly about Secondary storage devices? (JAN 2011)

The high-speed storage devices are very expensive and hence the cost per bit of storage is also very high. Again the storage capacity of the main memory is also very limited. Often it is necessary to store hundreds of millions of bytes of data for the CPU to process. Therefore additional memory is required in all the computer systems. This memory is called *auxiliary memory* or *secondary storage*.

In this type of memory the cost per bit of storage is low. However, the operating speed is slower than that of the primary storage. Huge volume of data are stored here on permanent basis and transferred to the primary storage as and when required. Most widely used secondary storage devices are *magnetic tapes* and *magnetic disk*.

1. **Magnetic Tape:** Magnetic tapes are used for large computers like mainframe computers where large volume of data is stored for a longer time. In PC also you can use tapes in the form of cassettes. The cost of storing data in tapes is inexpensive. Tapes consist of magnetic materials that store data permanently. It can be 12.5 mm to 25 mm wide plastic film-type and 500 meter to 1200 meter long which is coated with magnetic material. The deck is connected to the central processor and information is fed into or read from the tape through the processor. It similar to cassette tape recorder.

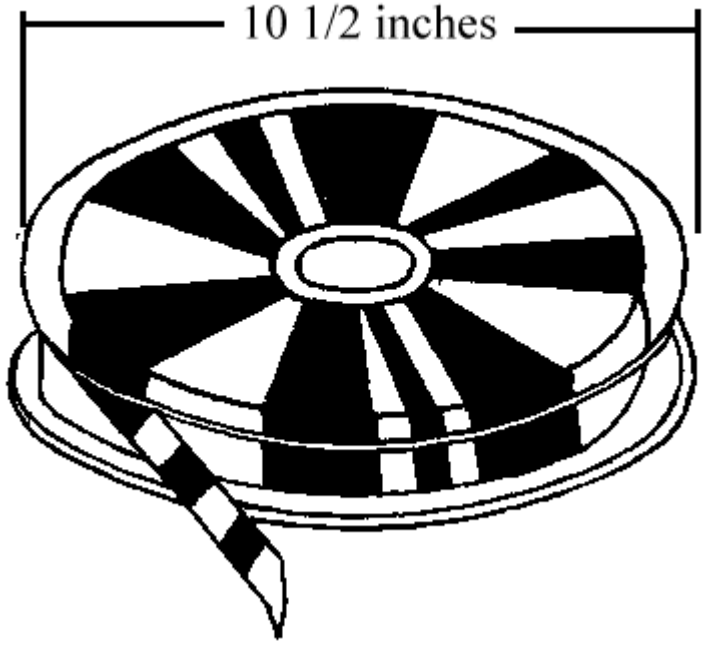


Fig. Magnetic Tape

Advantages of Magnetic Tape:

- **Compact:** A 10-inch diameter reel of tape is 2400 feet long and is able to hold 800, 1600 or 6250 characters in each inch of its length. The maximum capacity of such tape is 180 million characters. Thus data are stored much more compactly on tape.
- **Economical**: The cost of storing characters is very less as compared to other storage devices.
- **Fast**: Copying of data is easier and fast.
- **Long term Storage and Re-usability**: Magnetic tapes can be used for long term storage and a tape can be used repeatedly with out loss of data.

2. **Magnetic Disk:** You might have seen the gramophone record, which is circular like a disk and coated with magnetic material. Magnetic disks used in computer are made on the same principle. It rotates with very high speed inside the computer drive. Data is stored on both the surface of the disk. Magnetic disks are most popular for *direct access* storage device. Each disk consists of a number of invisible *concentric circles* called *tracks*. Information is recorded on tracks of a disk surface in the form of tiny magnetic spots. The presence of a magnetic spot represents *one bit* and its absence represents zero bit. The information stored in a disk can be read many times without affecting the stored data. So the reading operation is non-destructive. But if you want to write a new data, then the existing data is erased from the disk and new data is recorded.
3. **Floppy Disk:** It is similar to magnetic disk discussed above. They are 5.25 inch or 3.5 inch in diameter. They come in single or double density and recorded on one or both surface of the diskette. The capacity of a 5.25-inch floppy is 1.2 mega bytes whereas for 3.5 inch floppy it is 1.44 mega bytes. It is cheaper than any other storage devices and is portable. The floppy is a low cost device particularly suitable for personal computer system.
4. **Optical Disk:** With every new application and software there is greater demand for memory capacity. It is the necessity to store large volume of data that has led to the development of optical disk storage medium. Optical disks can be divided into the following categories:

1. ***Compact Disk/ Read Only Memory (CD-ROM)*:** CD-ROM disks are made of reflective metals. CD-ROM is written during the process of manufacturing by

high power *laser beam.* Here the storage density is very high, storage cost is very low and access time is relatively fast. Each disk is approximately 4 1/2 inches in diameter and can hold over 600 MB of data. As the CD-ROM can be *read only* we cannot write or make changes into the data contained in it.

2. ***Write Once, Read Many (WORM)*:** The inconvenience that we can not write any thing in to a CD-ROM is avoided in WORM. A WORM allows the user to write data permanently on to the disk. Once the data is written it can never be erased without physically damaging the disk. Here data can be recorded from keyboard, video scanner, OCR equipment and other devices. The advantage of WORM is that it can store vast amount of data amounting to gigabytes (10^9 bytes). Any document in a WORM can be accessed very fast, say less than 30 seconds.
3. ***Erasable Optical Disk*:** These are optical disks where data can be written, erased and re-written. This also applies a laser beam to write and re-write the data. These disks may be used as alternatives to traditional disks. Erasable optical disks are based on a technology known as *magnetic optical* (MO). To write a data bit on to the erasable optical disk the MO drive's laser beam heats a tiny, precisely defined point on the disk's surface and magnetises it.

7.Explain about memory in Computer System? (JAN 2011)

The Role of Memory

The term applies to any electronic component capable of temporarily storing data. There are two main categories of memories:

Internal memory that temporarily memorizes data while programs are running. Internal memory uses micro conductors, i.e. fast specialized electronic circuits. Internal memory corresponds to what we call random access memory (RAM).

Auxiliary memory (also called physical memory or external memory) that stores information over the long term, including after the computer is turned off. Auxiliary memory corresponds to magnetic storage devices such as the hard drive, optical storage devices such as CD-ROMs and DVD-ROMs, as well as read-only memories.

Technical Characteristics

(a) Capacity, representing the global volume of information (in bits) that the memory can store.

(b) Access time, corresponding to the time interval between the read/write request and the availability of the data.

(c) Cycle time, representing the minimum time interval between two successive accesses.

(d) Throughput, which defines the volume of information exchanged per unit of time, expressed in bits per second.

(e) Non-volatility, which characterizes the ability of a memory to store data when it is not being supplied with electricity.

The fastest memories are located in small numbers close to the processor. Auxiliary memories, which are not as fast, are used to store information permanently.

Types of Memories

Random Access Memory Random access memory, generally called RAM is the system's main memory, i.e. it is a space that allows you to temporarily store data when a program is running. Unlike data storage on an auxiliary memory such as a hard drive, RAM is volatile, meaning that it only stores data as long as it supplied with electricity. Thus, each time the computer is turned off, all the data in the memory are irremediably erased.

Read-Only Memory Read-only memory, called ROM, is a type of memory that allows you to keep the information contained on it even when the memory is no longer receiving electricity. Basically, this type of memory only has read-only access. However, it is possible to save information in some types of ROM memory. Flash Memory Flash memory is a compromise between RAM-type memories and ROM memories. Flash memory possesses the non-volatility of ROM memories while providing both read and writes access However, the access times of flash memories are longer than the access times of RAM.

The ideal memory has a large capacity with restricted access time and cycle time, a high throughput and is non-volatile. However, fast memories are also the most expensive. This is why memories that use different technologies are used in a computer, interfaced with each other and organised hierarchically.

8.Elaborate the various Input and Output Devices? (JAN 2011)

A computer is only useful when it is able to communicate with the external environment. When you work with the computer you feed your data and instructions through some devices to the computer. These devices are called Input devices. Similarly computer after processing, gives output through other devices called output devices.

Input Devices

Input devices are necessary to convert our information or data in to a form which can be understood by the omputer. A good input device should provide timely, accurate and useful data to the main memory of the computer for processing followings are the most useful input devices.

1. **Keyboard: -** This is the standard input device attached to all computers. The layout of keyboard is just like the traditional typewriter of the type QWERTY. It also contains some extra command keys and function keys. It contains a total of 101 to 104 keys. A typical keyboard used in a computer is shown in Fig. 2.6. You have to press correct combination of keys to input data. The computer can recognise the electrical signals corresponding to the correct key combination and processing is done accordingly.

2. **Mouse:** - Mouse is an input device shown in Fig. 2.7 that is used with your personal computer. It rolls on a small ball and has two or three buttons on the top. When you roll the mouse across a flat surface the screen censors the mouse in the direction of mouse movement. The cursor moves very fast with mouse giving you more freedom to work in any direction. It is easier and faster to move through a mouse.
3. **Scanner:** The keyboard can input only text through keys provided in it. If we want to input a picture the keyboard cannot do that. Scanner is an optical device that can input any graphical matter and display it back. The common optical scanner devices are Magnetic Ink Character Recognition (MICR), Optical Mark Reader (OMR) and Optical Character Reader (OCR).

    - **Magnetic Ink Character Recognition (MICR):** - This is widely used by banks to process large volumes of cheques and drafts. Cheques are put inside the MICR. As they enter the reading unit the cheques pass through the magnetic field which causes the read head to recognise the character of the cheques.
    - **Optical Mark Reader (OMR):** This technique is used when students have appeared in objective type tests and they had to mark their answer by darkening a square or circular space by pencil. These answer sheets are directly fed to a computer for grading where OMR is used.

    - **Optical Character Recognition (OCR):** - This technique unites the direct reading of any printed character. Suppose you have a set of hand written characters on a piece of paper. You put it inside the scanner of the computer. This pattern is compared with a site of patterns stored inside the computer. Whichever pattern is matched is called a character read. Patterns that cannot be identified are rejected. OCRs are expensive though better the MICR.

Output Devices

1. **Visual Display Unit:** The most popular input/output device is the Visual Display Unit (VDU). It is also called the monitor. A Keyboard is used to input data and Monitor is used to display the input data and to receive massages from the computer. A monitor has its own box which is separated from the main computer

system and is connected to the computer by cable. In some systems it is compact with the system unit. It can be *color* or *monochrome*.

2. **Terminals:** It is a very popular interactive input-output unit. It can be divided into two types: hard copy terminals and *soft copy* terminals. A *hard copy* terminal provides a printout on paper whereas soft copy terminals provide visual copy on monitor. A terminal when connected to a CPU sends instructions directly to the computer. Terminals are also classified as dumb terminals or intelligent terminals depending upon the work situation.
3. **Printer:** It is an important output device which can be used to get a printed copy of the processed text or result on paper. There are different types of printers that are designed for different types of applications. Depending on their speed and approach of printing, printers are classified as *impact* and *non-impact* printers. Impact printers use the familiar typewriter approach of hammering a typeface against the paper and inked ribbon. *Dot-matrix printers* are of this type. Non-impact printers do not hit or impact a ribbon to print. They use electro-static chemicals and ink-jet technologies. *Laser printers* and *Ink-jet printers* are of this type. This type of printers can produce color printing and elaborate graphics.

9.Illustrate the process of addition and subtraction in 1s and 2s complement system with suitable examples. (JAN 2012)

1's Complement Arithmetic

The Formula

$N = (2_n - 1) - N$

where: n is the number of bits per word

N is a positive integer

N is -N in 1's complement notation

For example with an 8-bit word and N = 6, we have:

$N = (2_8 - 1) - 6 = 255 - 6 = 249 = 11111001_2$

In Binary

An alternate way to find the 1's complement is to simply take the bit by bit complement of the binary number. For example: $N = +6 = 00000110_2$

$N = -6 = 11111001_2$

, given the 1's complement we can find the magnitude of the number by taking it's 1's complement.

The largest number that can be represented in 8-bit 1's complement is 01111111_2 = 127 = $7F. The smallest is 10000000_2 = -127. Note that the values 00000000_2 and 11111111_2 both represent zero.

Addition

End-around Carry. When the addition of two values results in a carry, the carry bit is added to the sum in the rightmost position. There is no **overflow** as long as the magnitude of the result is not greater than 2_n-1.

2's Complement Arithmetic

The Formula

$N N * = 2_n -$

where: n is the number of bits per word

N is a positive integer

N* is -N in 2's complement notation

For example with an 8-bit word and N = 6, we have:

$N* = 2_8 - 6 = 256 - 6 = 250 = 11111010_2$

In Binary

An alternate way to find the 2's complement is to start at the right and complement each bit to the left of the first "1".

For example: $N = +6 = 00000110_2$

$N* = -6 = 11111010_2$

Conversely, given the 2's complement we can find the magnitude of the number by taking it's 2's complement. The largest number that can be represented in 8-bit 2s complement is 01111111_2 = 127. The smallest is 10000000_2 = -128.

Addition

When the addition of two values results in a carry, the carry bit is ignored. There is no **overflow** as long as the is not greater than 2_n-1 nor less than -2_n.

10. With examples illustrate the conversion of decimal to Binary, octal and hexa decimal numbers. (Jan 2012)

Number System Conversion

- **Binary to Decimal**

$$X = 1101.01_2$$

$$\begin{aligned} Val(X) &: 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} \\ &: (8 + 4 + 0 + 1 + 0 + 0.25)_{10} = 13.25_{10} \end{aligned}$$

- **Decimal to Binary, Octal, Hexadecimal**

$$41.6875_{10} = (\ ?\ )_2$$

Example:

	b_6	b_5	b_4	b_3	b_2	b_1	b_0	b_{-1}	b_{-2}	b_{-3}	b_{-4}
Power	6	5	4	3	2	1	0	-1	-2	-3	-4
Value represented	64	32	16	8	4	2	1	0.5	0.25	0.125	0.0625
Enough? (*)	0	1	0	1	0	0	1.	1	0	1	1

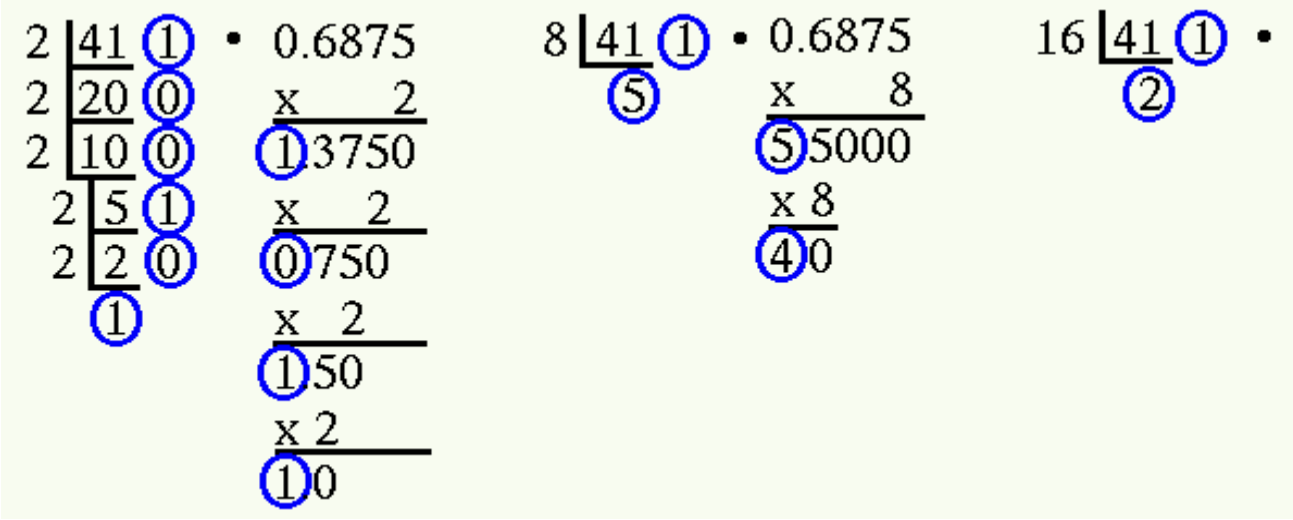
*

$41 - 32 = 9;\ \ 9 - 8 = 1;\ \ 0.6875 - 0.5 = 0.1875;\ \ 0.1875 - 0.125 = 0.0625$

So, we have

$$41.6875_{10} = 0101001.1011_2$$

A systematic method for number conversion



$41.6875_{10} = 101001.1011_2$

$41.6875_{10} = 51.54_8, \qquad 5 \times 8^1 + 1 \times 8^0 + 5 \times 8^{-1} + 4 \times 8^{-2} = 41.6875_{10}$

$41.6875_{10} = 29.A_{16} \qquad 2 \times 16^1 + 9 \times 16^0 + 11 \times 16^{-1} = 41.6875_{10}$

Given a value X in a certain number system, we want to convert it to a different system of radix r, i.e., we need to find all x_i's in the expression:

$$x_4x_3x_2x_1x_0\ .\ x_{-1}x_{-2}x_{-3}x_{-4}$$

The value of the integer part of the above expression is

$$val(X_{int}) = x_4r^4 + x_3r^3 + x_2r^2 + x_1r + x_0 = (((x_4r + x_3)r + x_2)r + x_1)r + x_0$$

Dividing this by r, we get x_0 as the remainder. If we keep dividing the previous quotient by r, the subsequent x_i's $(i = 1, 2, \cdots)$ can be found as the remainders.

Multiplying this by r, we get x_{-1} as the integer part of the product. If we keep multiplying the fraction part of the previous product by r, the subsequent x_i's $(i = -2, -3, \cdots)$ can be found as the integer part of the products.

Note: This method is mostly applicable to conversion of values given in decimal (base 10) system to other systems (base r=2, 8, 16, etc.), simply because we are trained to multiply and divide only in decimal system. In theorey one could use this method to convert a value given in any number system, but one has to know how to multiply and divide in **that** particular system.

More Examples:

$$425_{10} \longrightarrow (\cdots ? \cdots)_8$$
$$425_{10} \longrightarrow (\cdots ? \cdots)_{16}$$

- **Binary to Octal**

  Example:

$$1101011.1011_2 = \underline{001}\ \underline{101}\ \underline{011}.\underline{101}\ \underline{100}_2 = 153.54_8$$

- **Octal to Binary**

  Example: $27.35_8 = \underline{010}\ \underline{111}.\underline{011}\ \underline{101}_2 = 10111.011101_2$

- **Binary to Hexadecimal**

  Example: $1001011.1101 = \underline{0100}\ \underline{1011}.\underline{1101}_2 = 4B.D_{16}$

- **Hexadecimal to Binary**

  Example:

$$27.35_{16} = \underline{0010}\ \underline{0111}.\underline{0011}\ \underline{0101}_2$$

- **Binary-Coded-Decimal (BCD) Numbers**

$256_{10} = (0010\ 0101\ 0110)_{BCD}$

:

11. Convert the numbers: (JAN 2012)

(a) Convert the following number to decimal (i) (11011011.100101)2

(b) Convert (231.3)4 to Base of 7 (4)

(c) Convert the following Decimal numbers to Hexadecimal numbers

(i) (35)10, (ii) (275)10, (iii) (31)10

a)$11011011.100101_2=1\times2^7+1\times2^6+1\times2^4+1\times2^3+1\times2^1+1\times2^0.1\times2^{-1}+1\times2^{-4}+1\times2^{-6}=219.578125$

b)$231.3_4 = 45.75_{10} = 46_{10} = 64_7$

c) (i)$35_{10} = 23_{16}$

(ii)$275_{10} = 113_{16}$

(iii)$31_{10} = 1F_{16}$

12.Explain various generations of computer:

Each generation of computer is characterized by a major technological development that fundamentally changed the way computers operate, resulting in increasingly smaller, cheaper, more powerful and more efficient and reliable devices.

The history of computer development is often referred to in reference to the different generations of computing devices. Each generation of computer is characterized by a major technological development that fundamentally changed the way computers operate, resulting in increasingly smaller, cheaper, more powerful and more efficient and reliable devices. Read about each generation and the developments that led to the current devices that we use today.

First Generation (1940-1956) Vacuum Tubes

Sponsored
Migrate to a Private Cloud, Not a Virtual Datacenter: Before you implement a private cloud, find out what you need to know about automated delivery, virtual sprawl, and more.

The first computers used vacuum tubes for circuitry and magnetic drums for memory, and were often enormous, taking up entire rooms. They were very expensive to operate and in addition to using a great deal of electricity, generated a lot of heat, which was often the cause of malfunctions.

First generation computers relied on machine language, the lowest-level programming language understood by computers, to perform operations, and they could only solve one problem at a time. Input was based on punched cards and paper tape, and output was displayed on printouts.

The UNIVAC and ENIAC computers are examples of first-generation computing devices. The UNIVAC was the first commercial computer delivered to a business client, the U.S. Census Bureau in 1951.

Second Generation (1956-1963) Transistors

Transistors replaced vacuum tubes and ushered in the second generation of computers. The transistor was invented in 1947 but did not see widespread use in computers until the late 1950s. The transistor was far superior to the vacuum tube, allowing computers to become smaller, faster, cheaper, more energy-efficient and more reliable than their first-generation predecessors. Though the transistor still generated a great deal of heat that subjected the computer to damage, it was a vast improvement over the vacuum tube. Second-generation computers still relied on punched cards for input and printouts for output.

Second-generation computers moved from cryptic binary machine language to symbolic, or assembly, languages, which allowed programmers to specify instructions in words. High-level programming languages were also being developed at this time, such as early versions of COBOL

and FORTRAN. These were also the first computers that stored their instructions in their memory, which moved from a magnetic drum to magnetic core technology.

The first computers of this generation were developed for the atomic energy industry.

Third Generation (1964-1971) Integrated Circuits

The development of the integrated circuit was the hallmark of the third generation of computers. Transistors were miniaturized and placed on silicon chips, called semiconductors, which drastically increased the speed and efficiency of computers.

Instead of punched cards and printouts, users interacted with third generation computers through keyboards and monitors and interfaced with an operating system, which allowed the device to run many different applications at one time with a central program that monitored the memory. Computers for the first time became accessible to a mass audience because they were smaller and cheaper than their predecessors.

Fourth Generation (1971-Present) Microprocessors

The microprocessor brought the fourth generation of computers, as thousands of integrated circuits were built onto a single silicon chip. What in the first generation filled an entire room could now fit in the palm of the hand. The Intel 4004 chip, developed in 1971, located all the components of the computer—from the central processing unit and memory to input/output controls—on a single chip.

In 1981 IBM introduced its first computer for the home user, and in 1984 Apple introduced the Macintosh. Microprocessors also moved out of the realm of desktop computers and into many areas of life as more and more everyday products began to use microprocessors.

As these small computers became more powerful, they could be linked together to form networks, which eventually led to the development of the Internet. Fourth generation computers also saw the development of GUIs, the mouse and handheld devices.

Fifth Generation (Present and Beyond) Artificial Intelligence

Fifth generation computing devices, based on artificial intelligence, are still in development, though there are some applications, such as voice recognition, that are being used today. The use of parallel processing and superconductors is helping to make artificial intelligence a reality. Quantum computation and molecular and nanotechnology will radically change the face of computers in years to come. The goal of fifth-generation computing is to develop devices that respond to natural language input and are capable of learning and self-organization.

UNIT – II

1. Define Computer Software?

Software is a set of programs or collection of programs that is executed by the Computer's CPU to function it in a desired way.

2. What is meant by Installation and Assembling?

Installation –It is the process of loading the software package into the computer.

Assembling – It is the process of mounting different computer peripherals into one, to make the computer to function properly.

3. Difference between web page and website.

Web page	Web site
A **web page** is one screen full of information (from a web site) that may contain links to other pages in the web site or links to external information.	A **website** is a collection of one or more web pages designed to convey information on a particular subject or theme to a web user.
Every webpage should contain a Page Title in the head section. . A web page may contain text, animation or graphics elements.	Website will have a domain name. Example: A company will have a web site providing structured information about the company
Web page is an single entity.	Web site can have more than one web page.

4. What are the types of Software?

1. Application software , 2. System software.

5. Define OS.

An operating system is a set of programs, which are used to control and co-ordinate the computer system.

6. What are the basic functions of an OS?

Process Management, Memory Management, File Management, Device Management, Security Management, User Interface.

7. What are the types of Operating System? (JAN 2012)

Single user operating system, Multi-user operating system, Time sharing operating system, Virtual storage operating system, Real time operating system, Multiprocessing operating system, Virtual machine operating system.

8. Define Multiprocessing?

Multiprocessing is the process of executing a single job by using multiple CPU's.

9. What are language translators?

The language translators are the programs which come under system software category. They are Compilers, Interpreters and Assembler.

10. What are a Compiler, Assembler and Interpreter?

Compiler: It is a program which is used to convert the high level language program into machine language.

Assembler: It is a program which is used to convert the assembly level language program into machine language.

Interpreter: It is a program; it takes one statement of a high level language program, translates it into machine language instruction and then immediately executes the resulting machine language instruction.

11. What is Device Driver?

In computing, a device driver or software driver is a computer program allowing higherlevel computer programs to interact with a hardware device. A driver typically communicates with the device through the computer bus or communications subsystem to which the hardware connects.

12. What is the purpose of a Device Driver?

A device driver simplifies programming by acting as a translator between a hardware device and the applications or operating systems that use it. Programmers can write the higher-level application code independently of whatever specific hardware device it will ultimately control, because code and device can interface in a standard way, regardless of the software superstructure or of underlying hardware.

13. What is a linker?

A linker is a program that combines object modules to form an executable program. Many programming languages allow you to write different pieces of code, called modules, separately. This simplifies the programming task because you can break a large program into

small, more manageable pieces. Modules has to be put together. This is the job of the linker. In addition to combining modules, a linker also replaces symbolic addresses with real addresses.

14. What is a loader?

In computing, a loader is the part of an operating system that is responsible for one of the essential stages in the process of starting a program, loading programs, that is, starting up programs by reading the contents of executable into memory, then carrying out other required preparatory tasks, after which the program code is finally allowed to run and is started when the operating system passes control to the loaded program code.

15. What is Booting?

In computing, booting (also known as "booting up") is a bootstrapping process that starts operating systems when the user turns on a computer system. A boot sequence is the initial set of operations that the computer performs when power is switched on. The boot loader typically loads the main operating system for the computer.

16. What is application software?

An application software is a set of programs, that allows the computer to perform a specific data processing for the user.

17. How can you obtain required software?

Buying Pre-defined software, Buying customized software, Developing the software, Downloading from the Internet.

18. What are the categories of application software?

Customized Application Software, General Application Software.

19. Define the System.

System is a group of interrelated components working together towards a common goal.

20. Specify the personnel's, who are responsible for system design and implementation.

System Personnel, System Analyst, System Designer, Programmers, Users.

21. What is system development cycle?

System development cycle is the sequence of events considered by the system developers to build the new system or to replace the old one.

22. What are the phases of Software Development Cycle?

Requirement Analysis, Feasibility study, System Analysis and Design, Coding / Development, Testing, Implementation, Maintenance.

23. What is Software Requirement Specification (SRS) document?

The Software Required Specification (SRS) Document is produced at the end of Requirement Analysis stage, which specifies all requirements of the customer.

24. What is meant by Testing?

Testing is the process of executing the proposed software with sample or test data and put into regular use.

25. How the system can be tested?

Unit Testing, Integration Testing, System Testing, User Acceptance Test and Installation Testing.

26. Differentiate machine language and high level language.

Machine language	**High level Language**
Represented in numbers.	Human readable form.
Directly executed by the Central Processing Unit.	Should be translated into machine code by compiler / interpreter.
Example: ADD A, B where A and B are operands and ADD is an opcode.	**Example:** C, C++

27. Difference between Compiler and Interpreter.

Compiler	**Interpreter**
Executes source code into target or assembly code.	Executes source code directly or to an intermediate form.
Compilers convert once the source program.	Interpreter converts every time the program runs.
Languages for compiler conversion: C, C++.	Languages for interpreter conversion: MATLAB, Python.

28. What is IP Address?

Internet protocol address is the address of a device attached to an IP network (TCP/IP network). Every client, server and network device is assigned an IP address, and every IP packet traversing an IP network contains a source IP address and a destination IP address.

29. Name any four application software packages.

30. What is soft loading?

A soft reboot (also known as a warm reboot) is restarting a computer under software control, without removing power or (directly) triggering a reset line. It usually, though not always, refers to an orderly shutdown and restarting of the machine.

The Control-Alt-Delete key combination is used to allow the soft rebooting the system.

31. What are the steps involved in booting?

First, the Power On Self Tests (**POST**) is conducted. These tests verify that the system is operating correctly and will display an error message and/or output a series of **beeps** known as beep codes depending on the BIOS manufacturer.

Second, is initialization in which the BIOS look for the video card built in BIOS program and runs it. The BIOS then looks for other devices' ROMs to see if any of them have BIOSes and they are executed as well.

Third, is to initiate the boot process. The BIOS looks for boot information that is contained in file called the master boot record (MBR) at the first sector on the disk. If it is searching a floppy disk, it looks at the same address on the floppy disk for a volume boot sector. Once an acceptable boot record is found the operating system is loaded which takes over control of the computer.

32. What is the difference between text and graphical browsers?

Text browser	**Graphical browser**
No GUI	Based on GUI.
Links are based on text entry	Links are present as icons or images.
Allow users to display and interact only with text on the web pages.	Allow users to display and interact with various images, present on the web
Example: Lynx web browser	**Example:** Internet Explorer, Netscape Navigator.

33. What are the applications of software? (JAN 2011)

1. Operating system 2.Banking 3.Education 4.Entertainment

34. What is a web server? (JAN 2012)

Web servers are computers that deliver web pages. Every web server has an IP address and possibly a domain name. A **web server** is simply a computer program that dispenses web pages as they are requested..

PART – B

1. Give the categories of Software with example? (JAN 2011)

1. System software
2. Application software

SYSTEM SOFTWARE:

It is a collection of programs to perform system functionalities. They are of 2 types

1. System management program: divided into 3
   a) Operating system
   b) Utility programs
   c) Device drivers.
2. System development programs: divided in to
   a) Language translators
   b) Linkers
   c) Debuggers
   d) Editors

APPLICATION SOFTWARE

Classified in to standard application programs, unique application programs

Standard application programs divided into

a) Word processor
b) Spreadsheet
c) Database
d) Desk top publisher
e) Web browser

2. State different language translators and explain their functions?

Compiler and Loader:
(a)Compiler:

A compiler is a computer program (or set of programs) that transforms source code written in a programming language (the source language) into another computer language (the target language, often having a binary form knownasobjectcode).

(b)Loader:

In a computer operating system, a loader is a component that locates a given program (which can be an application or, in some cases, part of the operating system itself) in offline storage (such as a hard disk), loads it into main storage (in a personal computer, it is called random access memory), and gives that program control of the computer (allows it to execute its instructions). A program that is loaded may itself contain components that are not initially loaded into main storage, but can be loaded if and when their logic is needed. In a multitasking operating system, a program that is sometimes called a dispatcher juggles the computer processor time among different tasks and calls the loader when a program associated with a task is not already in main storage.

Linker:

Also called link editor and binder, a linker is a program that combines object modules to form an executable program. Many programming languages allow you to write different pieces of code, called modules, separately. This simplifies the programming task because you can break a large program into small, more manageable pieces. Eventually, though, you need to put all the modules together. This is the job of the linker. In addition to combining modules, a linker also replaces symbolic addresses with real addresses. Therefore, you may need to link a program even if it contains only one module.

The linkage editor accepts two major types of input:

1) Primary input, consisting of object decks and linkage editor control statements.

2) Additional user-specified input, which can contain both object decks and control statements, or load modules. This input is either specified by you as input, or is incorporated automatically by the linkage editor from a call library.

Output of the linkage editor is of two types:

1) A load module placed in a library (a partitioned data set) as a named member

2) Diagnostic output produced as a sequential data set.

3) The loader prepares the executable program in storage and passes control to it directly.

Interpreter:

An interpreter normally means a computer program that executes, i.e. performs, instructions written in a programming language. An interpreter may be a program that either

1) executes the source code directly,

2) translates source code into some efficient intermediate representation (code) and immediately executes this.

3) explicitly executes stored precompiled code made by a compiler which is part of the interpreter system.

Assembler:

An assembler translates an assembly language source program into machine codes. Though the assembly language is the symbolic representation of machine codes, a computer cannot

understand it. After translating the assembly language program into machine codes by the assembler, the program becomes ready for the execution.

3. **Explain in detail the steps involved in Software Development Process? (JAN 2012) (JAN 2011)**

Software development life cycle model is also called as waterfall model which is followed by majority of systems. This software development life cycle process has the following seven stages in it namely:

1.SystemRequirementsAnalysis

2.Feasibilitystudy

3.SystemsAnalysisandDesign

4.CodeGeneration

5.Testing

6.Maintenance

7. Implementation

System Requirements Analysis:
The first essential or vital thing required for any software development is system. Also the system requirement may vary based on the software product that is going to get developed. So a careful analysis has to be made about the system requirement needed for the development of the product. After the analysis and design of the system requirement phase the system required for the development would be complete and the concentration can be on the software development process.

Feasibility study:
After making an analysis in the system requirement the next step is to make analysis of the software requirement. In other words feasibility study is also called as software requirement analysis. In this phase development team has to make communication with customers and make analysis of their requirement and analyze the system. By making analysis this way it would be possible to make a report of identified area of problem. By making a detailed analysis on this area a detailed document or report is prepared in this phase which has details like project plan or schedule of the project, the cost estimated for developing and executing the system, target dates for each phase of delivery of system developed and so on. This phase is the base of software development process since further steps taken in software development life cycle would be based on the analysis made on this phase and so careful analysis has to be made in this phase.

Systems Analysis and Design:
This is an important phase in system development .Here analysis is made on the design of the system that is going to be developed. In other words database design, the design of the architecture chosen, functional specification design, low level design documents, high level design documents and so on takes place. Care must be taken to prepare these design documents because the next phases namely the development phase is based on these design documents. If a well structured and analyzed design document is prepared it would reduce the time taken in the coming steps namely development and testing phases of the software development life cycle.

Code Generation:

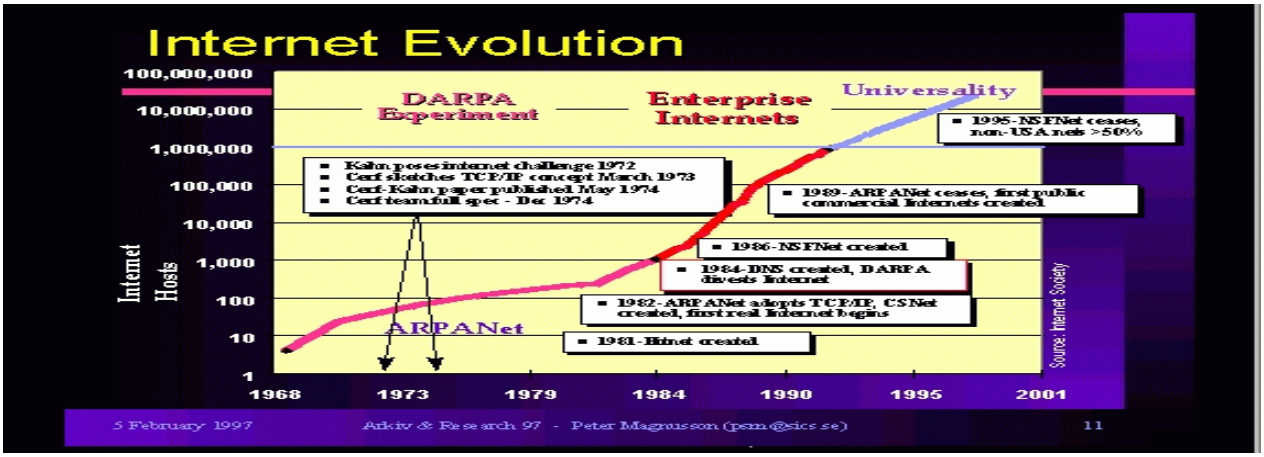
This is the phase where actual development of the system takes place. That is based on the design documents prepared in the earlier phase code is written in the programming technology chosen.

After the code is developed generation of code also takes place in this phase. In other words the code is converted into executables in this phase after code generation.

Testing:

A software or system which is not tested would be of poor quality. This is because this is the phase where system developed would be tested and reports are prepared about bugs or errors in system. To do this testing phase there are different levels and methods of testing like unit testing, system test and so on. Based on the need the testing methods are chosen and reports are prepared about bugs. After this process the system again goes to development phase for correction of errors and again tested. This process continues until the system is found to be error free. To ease the testing process debuggers or testing tools are also available.

4. Write a short note on evolution of Internet?



First experiments in 1966

ARPA requested quotations in 1968

Packet switching was widely doubted, but was desired for its fault tolerance. One objective was to build a system that could withstand any systematic attack on central nodes, such as from a nuclear strike.

(The van Johnson story)

AT&T was particularly pessimistic!

4 computers on ARPANET in 1969

in 1973 an effort began to connect ARPANET with mobile networks using synchronous satellites (SATNET) and mobile packet radio (PRNET), this effort became known as internetting

Part of this research effort resulted in Ethernet (in Hawaii)

July 1977 a four-network demonstration linked ARPANET, SATNET, and the PRNET. TCP/IP version 4 came in 1978.

the new internet protocols, generally called TCP/IP, began to be developed in the early 70s, and ARPANET switched to them in January 1983

supercomputer centers programmed in 1986 (Senator Goreĺs legislation) led to the NFSNET, which remained the backbone until April 1995

experimental electronic mail relay put into operation in 1989, interconnecting MCI Mail with the Internet. Compuserve, ATTMail, and Sprintmail followed shortly

Principal Functions

- e-mail
- hobbies
- news
- personal publishing
- product information
- software distribution
- commerce
- telephony

According to (ftp://ftp.nw.com/zone), in january 96 there were 9.5 million hosts. Growth rate is 20% per quarter.

Quantitatively, the success of the Internet is obvious. Less obvious is that the growth has been fairly stableThe WWW was conceived by Tim Berners-Lee in March 1989, in a CERN proposal. It first aimed at managing information about accelerators and experiments at CERN.

The concept of ôwebö derived from an observation of how new staff at CERN were introduced to the work structure - by being given a few hints on who to talk to. At CERN there is and was a large turnover, several thousand people are involved yet seldom stay longer than 2 years.

Also improves on problems with hierarchical data structures (such as file systems) and keyword based systems (such as bibliographic databases).

Other than using TCP/IP, a key idea is a uniform naming scheme that includes the service name. Also, in HTTP the client can send a list of the representations it understands, and the server reply appropriately.

NCSA Mosaic came in 1992, and was instantly popular. Netscape Communications was formed in 1994.

There is not really any new technology in the Web concept, just a good combination of ideas.

Matthew Gray used the first web ôspiderö, that he wrote in the spring of 1993, to locate as many web servers as possible.

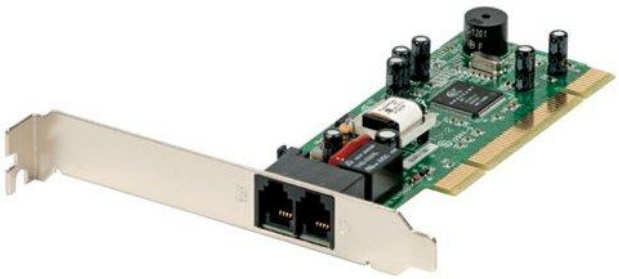
Today there are over internetwork 600000 web servers (january 1997), according to Netcraft.

The dotted Lines are my own estimates based on Netcraft and Internet society data.

5.Explain various types of Internet Connections?

Types of Internet Connections.

As technology grows, so does our need for things to go faster. Ten years ago, websites just included images, coloured text and some repetitive melodies. Now Flash websites, animations, high resolution photos, online gaming, videos or streaming (radio on the internet), are getting more popular for people who demand faster and faster internet connections.
The connection speeds listed below represent an average speed at the time of publication (May 2009). This will no doubt change over time.

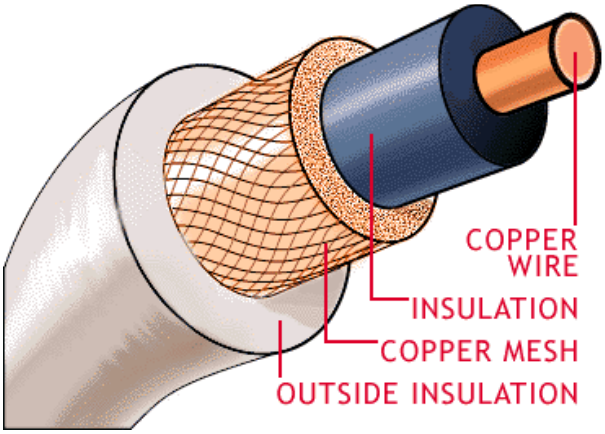


1) **PCI modem**(see image above). Analogue up to 56000 bits per second. It means that in a second, 56000 bits (0 or 1) travel through the copper wire. It is both economical and slow and it is also called dial-up access. If you connect the modem, you get internet but as it uses the analogue telephone line, if you surf on the internet, nobody can call you because the line is busy.
Using a modem connected to your PC which is very cheap (about 10 €) , users connect to the Internet only if you click on the telephone Access Icon and the computer dials the phone number provided by your ISP (Internet Service Provider) and connects to the network. The signal is analogue because data is sent over an analogue telephone network. This modem converts received analogue data to digital (always analogue on the

telephone site and digital on the computer side).
As dial-up access uses ordinary telephone lines the data rates are limited and the quality of the connection is not always good. Nowadays very few people use this type of connection.

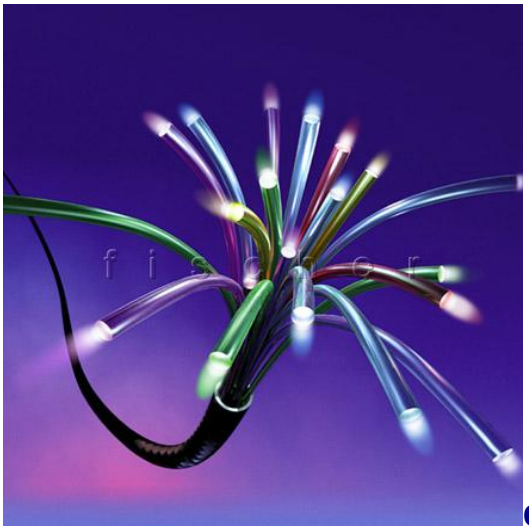
2) **DSL**

DSL or - an 'always on' connection- uses the existing 2-wire copper telephone line connected to the internet and won't tie up your phone like the old modem does. There is no need to dial-in to your ISP as DSL is always on. DSL is called ADSL (Short for Asymmetric Digital Subscriber Line) for home subscribers.
As we said before ADSL is short for asymmetric digital subscriber line and supports data rates up to 10Mbits (May 2009) when receiving data (download) and from 16 to 640 Kbps when sending data (upload). ADSL is called asymmetric because it supports different data rates for upload than for download traffic.



3) **Cable**

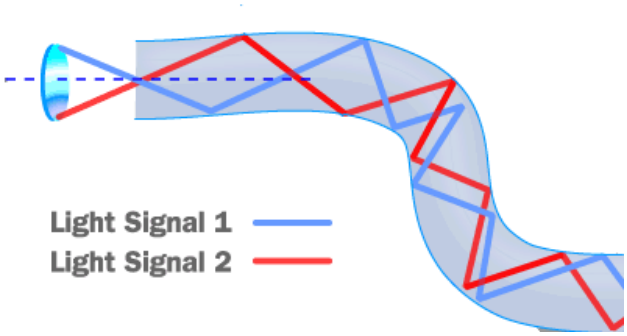
There are two type of cable; Coaxial and optic fibre. The first one is used by cable TV and that is common for data communications (see image on the left).
The cross-section of the cable shows a single centre solid wire made of copper surrounded by a copper mesh conductor. Between the main wire (in the centre) and the mesh conductor is an insulating dialectric. This dialectric (blue part in the image) has a large effect on the essential features of the cable. Depending on the material that isulator is made of, the cable has different inductance and capacitance values and these values affect how quickly data travels through the wire. The last layer is an outside insulator to protect the whole wire.
Data is transmitted through the rigid wire, while the outer copper mesh layer serves as a line to ground.



Optic Fibre.

Fibre-optic cables are strands of a special optical material as thin as a human hair that carry data (files, videos ..) over long distances. Now, there is not electrical signal. In Optical fibres data are carried as light signals

How Does an Optical Fiber Transmit Light?

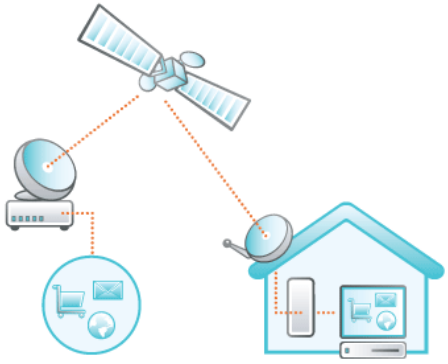


What is the secret of optical Fibre? Why doesn't the light ray escape from the strand? Suppose you want to shine a torch beam down a long, straight corridor. Just point the beam straight down the corridor. -- light moves in straight lines so the light will reach to the end of the corridor.
What if the corridor has a bend in it? . Just place a mirror at the bend to reflect the light beam towards the other side of the corridor.
What if the corridor has multiple bends? You might places as many mirrors as bends so that it bounces from side-to-side all along the corridor. This is what happens in an optical fibre.

4) Wireless Internet Connections

Wireless broadband (Wireless Internet Connections). Instead of using cable networks for your Internet connection, WIC uses radio frequency .Wireless Internet can be accessed from anywhere as long as your WIFI adaptor is located within a network coverage area. It also provides an always-on connection and it is still considered to be relatively new.

5) **Satellite**



IoS short for Internet over Satellite allows a user to access the Internet via a geostationary satellite that orbits the earth. A geostationary satellite is a type of satellite placed at a fixed position above the earth's surface. Because of the large distances between home and satellite, signals must travel from the earth up to the satellite and back again. It causes a slight delay between the request and the answer.

6.**Explain the functions of operating system. (JAN 2012)**

Application software

-Easy-to-use programs designed to perform specific tasks.

System software

It is a collection of programs that controls and manage the computer.
Example: OS, Language processor etc,.

OPERATING SYSTEM

It is a collection of programs that controls and manages the computer
Examples : Windows, Unix, MSDOS

Functions Of OS

- It provide an interface between the hardware and the user.
- It controls and co-ordinate the entire computer system.
- It controls the allocation and use of various resource by various user and task.
- It controls the various application programs.
- Scheduling the jobs.
- Process management, Memory management, Device management etc,.

Device drivers

- It is set of programs, which act as an interface between the computer and the device.
- It is responsible for the proper functioning of the device.

Language Processor

- It is a system s/w that translates the programs written in High level language to Machine language.
- Machine language: 0's and 1's.
- High level language: C, C++, Java etc,.
  Example: Compiler, Interpreter, Assembler

Compiler

- It converts the programs written in high level language to machine language i.e. it translates the source code to object code.

Interpreter

- It converts the programs written in high level language to machine language.
- It executes the source code in line-by-line manner

Assembler
It converts the programs written in Assembly language to machine language

Application software

- **Customised Application s/w**
  It is developed to meet the requirements of limited user. Example Ms-Office, Reservation system,
  Payroll processing system
  Hospital management system etc,.

- **General Application s/w**
  It is developed to meet the requirements of many user.

7. Define various Internet Terminologies?

Internet - Network of millions of computers used to send information back and forth to one another. There is no authority agency or company that decides what can and cannot be published on the internet. This means that virtually anyone can publish, and means that there are large numbers of sites that contain false and misleading information. It is important to check who takes credit for the site you are viewing.

Homepage - Default setting, or the page that opens up when you start your Browser. A homepage can also be the entry point for people viewing information that is provided.

Links - Hypertext which, when clicked, connects you to another site or another page within the same site. In most cases, these links are usually highlighted in blue and are always underlined.

Page - What is on your screen at any given time. The page includes all the information available to you by scrolling up or down; however, when you click on hypertext your computer will 'jump' to a new page.

Search Engines - Software that allows you to search the Web by typing in a topic of interest.

Examples of search engines we use are Google, HotBot, and Northern Light. These search engines find exact matches from what has been typed in the search screen to either documents (files) or subjects of files on the WWW.

URL or Uniform Resource Locator - The address for a specific file on the web, as well as a method for directing users to a specific file, at a specific site. To type in a different URL, press command and L at the same time on a Macintosh computer, control and L on a PC, or locate the address bar at the top of the screen, type in the URL, and then press return. The URL for the Blaine School District Home page is: http://www.blaine.k12.wa.us.

Web Browser - To access the WWW, you need a Browser. A Web Browser is client software that gets information from a server. It interprets the information, formats it, and displays it on your computer screen. The most popular browser, and the one we use most often in Blaine, is Internet Explorer.

WWW, World Wide Web or Web - Interactive collection of hypertext pages linked to one another. They may include text, graphics and/or links to other spots in the Web. The Web is interactive because a user can click on text or graphics to navigate (move) to more information or other graphics, sounds and video that is of interest to them.

Home page - Generally the first page retrieved when accessing a Web site. Usually a "home" page acts as the starting point for a user to access information on the site. The "home" page usually has some type of table of contents for the rest of the site information or other materials. When creating Web pages, the "home" page has the filename "index.html," which is the default name. The "index" page automatically opens up as the "home" page.

HTML - A type of text code in Hypertext Markup Language which, when embedded in a document, allows that document to be read and distributed across the Internet

HTTP - The hypertext transfer protocol (http) that enables html documents to be read on the Internet.

IP Address - (Internet Protocol) The number or name of the computer from which you send and receive information on the Internet.

Web server: A computer program that is responsible for accepting HTTP requests from web clients, which are known as web browsers, and serving them HTTP responses along with optional data contents, which usually are web pages such as HTML documents and linked objects (images, etc.).

8. Explain various Internet applications, advantages and drawbacks? (JAN 2012)

In the history of mankind, the Internet is the greatest development in the domain of communication industry. Similar to each and every invention, the Internet carries a number of advantages and disadvantages. Nevertheless, the advantages of the Internet are so huge in number that they outperform the disadvantages quite easily.

According to the definition provided by Oxford dictionary, the Internet is an arrangement of connected computers, which lets the computer users all over the globe exchange data. At the present time, approximately 33% of the world population has accessibility to the Internet. The Internet is an extraordinary entertainment and learning tool that may be utilized in a number of modes to increase the ability of a user to collect information. The principal components of the Internet are the World Wide Web (WWW) and e-mail. With the passage of time, the Internet has become the most effective business tool in the contemporary world. It can be described as a global meeting place where people from every corner of the world can come simultaneously.

The advantages of Internet

Following are the advantages provided by the Internet:

1)Information The biggest benefit offered by the Internet is information. It functions as a valuable resource of information. You can find any type of information on any subject with the help of the search engines like Yahoo and Google.
2) Communication The primary goal of the Internet is communication. It has done extremely well in this field, however the development process is still going on to make it more dependable and quick. By sending an e-mail, we can contact a person who is physically present thousand miles away within the fraction of a second's time.
3) Entertainment Internet functions as a popular medium of entertainment. A wide variety of entertainment including video games, music, movies, chat room, news and others can be accessed through the Internet.
4) E-commerce E-commerce is the idea that is implemented for any form of commercial strategy or business transactions that entails transmission of data from one corner of the world to another. E-commerce has become a fantastic option through which you can shop anything.

5) Formation of communities Internet helps in formation of communities or forums. Here a number of people can participate in different types of debates and discussions, express their views and gather valuable knowledge.
6) Services A variety of services are offered via Internet, for example job searching, online banking, buying movie tickets, hotel reservations and consultation services etc. When you avail these services offline, they become more expensive.

The disadvantages of Internet

Following are the disadvantages of Internet:

1) Spamming: Spamming denotes distribution of unsolicited e-mails in large numbers. They are meaningless and they unnecessarily block the whole system. These activities are treated as illegal.
2) Theft of personal details While using the Internet, there is high probability that your personal details like name, address and credit card number may be accessed by con artists and used for fraudulent purposes.
3) Pornography: Pornography is definitely harmful for your children. There are numerous pornographic sites available over the Internet and watching any of those can have very bad influence on the mental health of your children.
4) Virus threat Virus is a program that interrupts the usual operation of your personal computer system. PCs linked to the Internet have high probability of virus attacks and as a result of this your hard disk can crash, giving you a lot of 1) Information The biggest benefit offered by the Internet is information. It functions as a valuable resource of information. You can find any type of information on any subject with the help of the search engines like Yahoo and Google.
2) Communication The primary goal of the Internet is communication. It has done extremely well in this field, however the development process is still going on to make it more dependable and quick. By sending an e-mail, we can contact a person who is physically present thousand miles away within the fraction of a second's time.
3) Entertainment Internet functions as a popular medium of entertainment. A wide variety of entertainment including video games, music, movies, chat room, news and others can be accessed through the Internet.
4) E-commerce E-commerce is the idea that is implemented for any form of commercial strategy or business transactions that entails transmission of data from one corner of the world to another. E-commerce has become a fantastic option through which you can shop

anything.
trouble.

Applications

a. Education
b. Entertainment
c. Research
d. Science
e. Medicine
f. Communication
g. Games

9.Brief the major problems encountered in the software system?

(a) Correctness:

The correctness of the software system refers to

1. Agreement of program code with specification.

2. Independence of the actual application of the software system.

(b) Reliability

Reliability of a software system is defined as the probability that this system fulfills a function for a specified number of input trails under specified input conditions in a specified time interval. Also if the test produces the lower error rate the system is reliable. The error rate depends on the frequency of inputs and on the probability that an individual input will lead to error.

(c) User Friendliness

Adequacy

1. The input required for the user should be limited to only what is necessary.

2. The performance offered by the software system should be adapted to extensibility.

3. The results that a software system delivers should be output in a clear and wellstructured form and be easy be interpret.

Learnability

1. The user interface should present more information as close to reality as possible and permit efficient utilization of the failures of the software.

2. The user manual should be structured clearly and simply.

(d) Robustness

A software system is robust if the consequences of an error in its operation, in the input, or in the hardware, in relation to a given application, and inversely proportional to the probability of the occurrence of this error in the given application.

(e) Maintainability

The maintainability of the software depends on 1. Readability

2. Extensibility

3. Testability (f) Readability

It depends on

1. Form of representation

2. Programming style

3. Consistency

4. Structure of the program

5. Programming language used

6. Quality of documentation

7. Tools available for inspection

(e) Extensibility

It depends on

1. Structure of the software system

2. Readability of the code

3. Availability of program documentation

4. Implementation program

(f) Testability

Allows to debugging of the program during execution. It depends on

Modularity: well structured programs suitable for stepwise testing

Structuredness: Useful for systematic testing of all components.

(g) Efficiency

Ability of a software system to fulfill its purpose with the best possible utilization of all necessary resources (time, storage, transmission, channels and peripherals).

(h) Portability

A software system which can be adapted to run on computers other than the one which it was designed.

It depends on

i. Degree of hardware independence

ii. Implementation language

iii. Specialized system functions

iv. Hardware properties

v. System dependent elements are collected in easily interchangeable program components.

10.Explain in detail about booting techniques.

Starting the Computer

Explain that the process of starting the computer is known as **booting**. Explain what happens when you start or restart a computer, and compare **cold boot** to **warm boot**. Discuss the role of the **kernel**, or **supervisor program**, in the booting process. Address why the kernel is called a **memory resident**, and explain what a **nonresident** is.

1. **The BIOS Screen and Setup Program.** Pronounce BIOS "bye-ose." Define BIOS as built-in software that determines what a computer can do without accessing programs from a disk. Because the BIOS is typically found on a ROM chip that comes with the computer, it is always available and cannot be lost due to a disk failure.

2. **The Power-On Self-Test.** Explain that this step is also called POST and is a diagnostic testing sequence run by the computer's BIOS. When everything checks out as it should, the computer will sound a reassuring beep. When errors are encountered, a series of beeps are sounded and an error message appears on the screen.

3. **Loading the OS.** Explain to students that once POST is successful, the BIOS finds and loads the operating system into RAM. Define **CMOS** (pronounced "cee-moss") as a type of nonvolatile memory that stores the date, time, and system setup parameters. Emphasize the importance of having an **emergency boot disk**.

4. **Configuring the System.** Once the OS is loaded**,** the system loads any utility programs it might need to run peripheral devices. Explain the term **plug-and-play** and how it can automatically detect compatible peripherals when the power is switched on. This might be a good time to define the term **registry** and inform students that this very important feature of Windows will be discussed in detail.

5. **Loading System Utilities.** Use Figure 5.2 to explain that the next step in the booting process is loading system utilities, which include all of the configuration choices found in the **Control Panel**.

6. **Authenticating Users.** The **authentication** or **login** process is the last step in starting the computer. Explain that consumer-oriented operating systems such as Windows and Mac OS do not demand that you supply a user name and password to use the computer. However, **profiles** can be used for that purpose. Multiuser computer systems require that you have an account to use the system. Explain that these accounts are managed by the **system administrator** or SysAdmin.

11.What are types of testing? Explain.

1. **Unit Testing**

(i) Unit Testing is primarily carried out by the developers themselves
(ii) Deals functional correctness and the completeness of individual program units
(iii) White box testing methods are employed

2. Integration Testing

(i) **Integration Testing:** Deals with testing when several program units are integrated
(ii) **Regression testing:** Change of behavior due to modification or addition is called 'Regression'. Used to bring changes from worst to least
(iii) **Incremental Integration Testing:** Checks out for bugs which encounter when a module has

been integrated to the existing
(iv) **Smoke Testing:** It is the battery of test which checks the basic functionality of program. If fails then the program is not sent for further testing

3. System Testing

(i) **System Testing :** Deals with testing the whole program system for its intended purpose
(ii) **Recovery testing:** System is forced to fail and is checked out how well the system recovers the failure
(iii) **Security Testing:** Checks the capability of system to defend itself from hostile attack on programs and data
(iv) **Load & Stress Testing:** The system is tested for max load and extreme stress points are figured out
(v) **Performance Testing:** Used to determine the processing speed
(vi) **Installation Testing:** Installation & uninstallation is checked out in the target platform

4. Acceptance Testing

(i) **UAT:** ensures that the project satisfies the customer requirements
(ii) **Alpha Testing :** It is the test done by the client at the developer's site
(iii) **Beta Testing :** This is the test done by the end-users at the client's site
(iv) **Long Term Testing :** Checks out for faults occurrence in a long term usage of the product
(v) **Compatibility Testing :** Determines how well the product is substantial to product transition

12.Describe how does one get connected to internet. (JAN 2012)

World Wide Web - It is a collection of information or collection websites.

E-mail - It is used to send electronic message to anyone.

Chatting

- It is online Conversation.
- It is used to send message back and forth to anyone.

Remote Access

- It is the process of accessing information present in a remote computer.

File Sharing

- It enables a group of user to share information.
- The information has been placed on a shared location and the user access the information.

IRC

- Internet Relay Chat.
- It is designed for group communication.

Video Conferencing

- It allows two or more user to interact via video and audio transmission.

FTP

- File Transfer Protocol
- It is used to transfer data from one computer to another computer.

TelNet

- Telecommunication Network.This protocol is used to get services from the server

VOIP

- Voice Over Internet Protocol.
- It sends voice over the internet protocol.

UNIT – III
PART – A

1. What is a program?

A program is a set instruction written to carryout a particular task, so that computer can perform some specified task.

2. What is algorithm?

Algorithm means the logic of a program. It is a step-by-step description of how to arrive at a solution of a given problem.

3. What are the steps to solve the problem in a computer system?

Problem must be analyzed thoroughly, Solution method is broken down into a sequence of small tasks, Based on this analysis, an algorithm must be prepared to solve the problem, The algorithm is expressed in a precise notation. This notation is known as "Computer Program", The Computer program is fed to the computer, The instruction in the program executes one after another and outputs the expected result.

4. How can you measure the quality of algorithm?

The primary factors that are often used to judge the quality of an algorithm are time requirement, memory requirement, and accuracy of solution.

5. What are the characteristics of an algorithm?

1. In algorithms each and every instruction should be precise, 2. In algorithms each and every instruction should be unambiguous, 3. The instructions in an algorithm should not be repeated infinitely, 4. Ensure that the algorithm will ultimately terminate, 5. The algorithm should be written in sequence, 6. It looks like normal English, 7. The desired result should be obtained only after the algorithm terminates.

6. How many types the Algorithm can be represented?

Normal English, Program, Flowchart, Pseudo code, Decision table

7. What is decision table?

A decision table is a table containing the selection of conditions to be tested and how those conditions should be nested to arrive at the proper action.

8. What is Flowchart?

A Flowchart is a pictorial representation of an algorithm. It is often used by programmer as a program planning tool for organizing a sequence of step necessary to solve a problem by a computer.

9. What is the need of Flowchart symbols?

Each symbol of different shapes denotes different shapes denote different types of instructions. The program logic through flowcharts is made easier through the use of symbol that has standardized planning.

10. What is pseudo code?

"Pseudo" means imitation of false and "code" refers to the instruction written in the programming language. Pseudo code is programming analysis tool that is used for planning program logic.

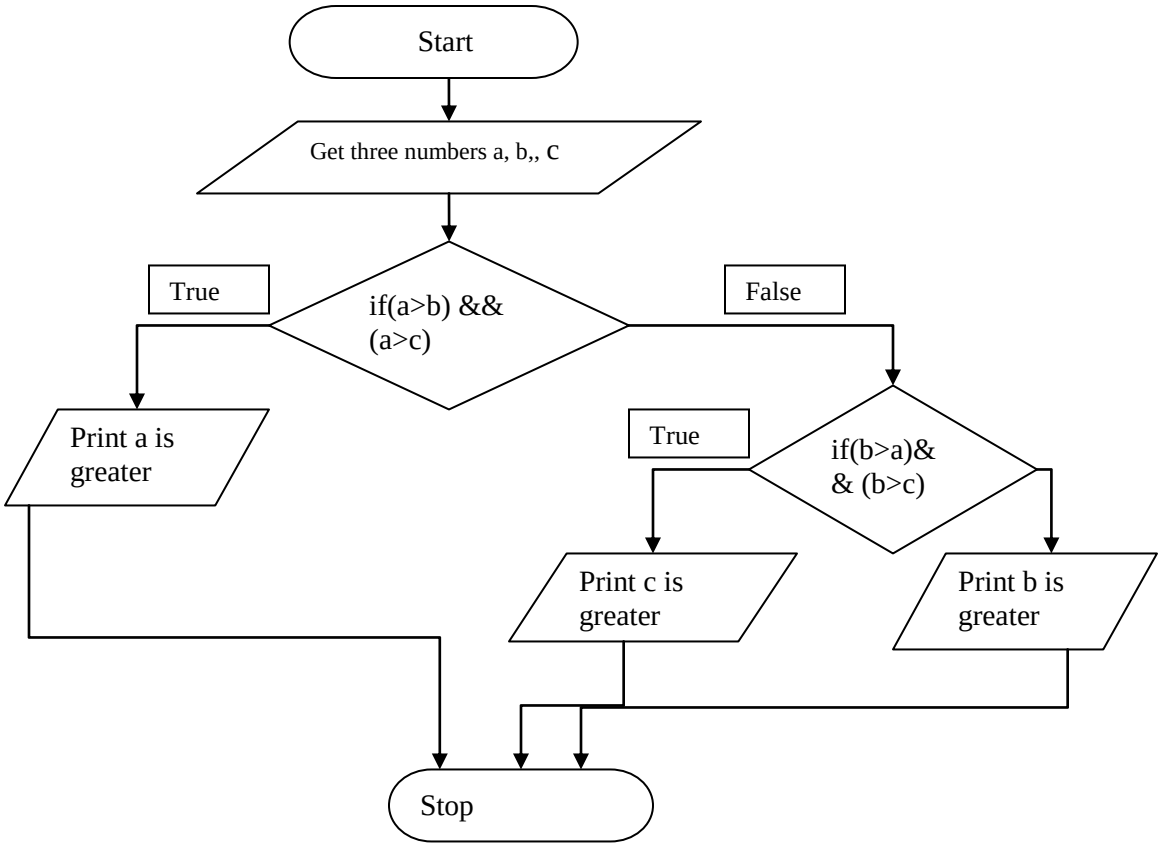
11. What is structured programming?

A structured programming is a more specific approach to solve a programming problem by using only the three basic logic structures. They are sequence logic, selection logic and Iteration logic.

12. What are the rules for drawing a flowchart?

The standard symbols should only be used, The arrowheads in the flowchart represent the direction of flow of control in the problem, The usual direction of the flow of procedure is from top to bottom or left to right, The flow lines should not cross each other, Be consistent in using names and variables in the flowchart, Keep the flowchart as simple as possible, Words in the flowchart symbols should be common statements and easy to understand, Chart main line of logic, and then incorporate all the details of logic, If a new page is needed for flowcharting, then use connectors for better representation, Don't chart every details or the flowchart will only be graphical represented.

13. Draw the flowchart to find the maximum among three numbers



14. What is sequence logic?

Sequence logic is used for performing instructions one after another in a sequence.

15. What is selection logic?

Selection logic is used for selecting the process path out of two or more alternative paths in the program logic. It uses three control structures called if…then, if… then…else and switch…case.

16. What is Iteration logic?

Iteration logic is used for producing loops in program logic when one or more instructions may be executed several times depending on some condition. It uses two control structures called do…while, and repeat…until.

17. What are the rules for writing pseudo code?

Write on statement per line, Capitalize initial keywords, Indent to show hierarchy, End multi line structure, Keep statements language independent.

18. What are the features of word processors? (JAN 2011)

Fast, Permanent storage, Formatting, Editing, Graphics, OLE, Spell Check, Mail merge

19. How many types a documented can be viewed?

Normal view , Online layout, Outline view , Page layout view

20. What are the menus available in Ms-Word?

File, Edit, View, Insert, Format, Tools, Table, Window, Help

21. What is meant by Formatting?

Formatting is the process of changing the appearance of the text in the document.

22. Specify any five toolbars available in Ms-Word?

Standard, Formatting, Drawing, Tables & Boarders, WordArt etc.,

23. How many Line Spacing options available in Ms-Word?

Single, 1.5 Lines, Double, At least, Exactly, Multiple

24. What are the Text cases available in Ms-Word?

Sentence case, lower case, UPPER CASE, Title Case, tOGGLE CASE

25. What is Subscript and Superscript?

The Subscript format places the text slightly below a line of normal printed text.

Eg:-H2O The Superscript format places the text slightly above a line of normal printed text.

Eg:-A2+B2

26. What is Tab and what are the Tab settings available in word?

Tab is used to control the alignment of text with in the document. Word provides

seven types of tabs.

Standard (left) tab, Center tab, Right tab, Decimal tab, Bar tab, First line Indent tab, Hanging Indent tab

27. Define Headers and Footers.

Header allows text, page number or section titles to appear on every page of document at the top position. Footer allows text, page number or section titles to appear on every page of document at the bottom position.

28. What is a Table?

A Table is grid of rows and columns.

29. What is a Clipart?

Clipart is the attractive pre-defined, pre-colored pictures available in Ms-Word office suite.

30. Define a Template.

Template is a special kind of document that produces basic tools for shaping a final document.

31. Enlist the advantage of algorithm. (JAN 2012)

The term algorithm is now applied to many kinds of problem solving that employ a mechanical sequence of steps, as in setting up a computer program. The sequence may be displayed in the form of a flowchart in order to make it easier to follow.

32. State the usage of Excel application software. (JAN 2012)

Microsoft Excel is a spreadsheet tool capable of performing calculations, analyzing data and integrating information from different programs. Microsoft Excel is comprised of organizational units called **workbooks**. A standard workbook contains **worksheets** and **chart sheets**. Worksheets perform calculations, store and organize data, present graphics and controls like a web page; they are extremely versatile. A worksheet in turn is comprised of millions of **cells**. The job of a cell is to store a **formula** that performs a calculation or communicates with some other application (i.e. program) such as a database.

PART – B

1. Explain the steps involved in developing a program with neat diagram?

It is the process of developing the software is called as SDLC. It is a cyclic process.

1. Problem definition: problem must be clearly defined by the user.
2. Analysis: what is to be solved? – collecting requirements
3. Design: Developing the procedure to solver the requirements.
4. Coding: Implementation of design procedures in programming language is called as coding.
5. Testing: perform checking the correctness of the program. It checks the symmetric errors. Classified into alpha testing & beta testing.
6. Deployment & Maintenance: Putting the software in real time use after testing is called as deployment.

Planning

The important task in creating a software product is extracting the requirements or requirements analysis. Customers typically have an abstract idea of what they want as an end result, but not what software should do. Incomplete, ambiguous, or even contradictory requirements are recognized by skilled and experienced software engineers at this point. Frequently demonstrating live code may help reduce the risk that the requirements are incorrect.

Once the general requirements are gleaned from the client, an analysis of the scope of the development should be determined and clearly stated. This is often called a scope

document. Certain functionality may be out of scope of the project as a function of cost or as a result of unclear requirements at the start of development. If the development is done externally, this document can be considered a legal document so that if there are ever disputes, any ambiguity of what was promised to the client can be clarified.

Design

Domain Analysis is often the first step in attempting to design a new piece of software, whether it be an addition to an existing software, a new application, a new subsystem or a whole new system. Assuming that the developers (including the analysts) are not sufficiently knowledgeable in the subject area of the new software, the first task is to investigate the so-called "domain" of the software. The more knowledgeable they are about the domain already, the less work required. Another objective of this work is to make the analysts, who will later try to elicit and gather the requirements from the area experts, speak with them in the domain's own terminology, facilitating a better understanding of what is being said by these experts. If the analyst does not use the proper terminology it is likely that they will not be taken seriously, thus this phase is an important prelude to extracting and gathering the requirements.

Specification

Specification is the task of precisely describing the software to be written, possibly in a rigorous way. In practice, most successful specifications are written to understand and fine-tune applications that were already well-developed, although safety-critical software systems are often carefully specified prior to application development. *Specifications are most important for external interfaces that must remain stable.* A good way to determine whether the specifications are sufficiently precise is to have a third party review the documents making sure that the requirements and Use Cases are logically sound.

Architecture

The architecture of a software system or software architecture refers to an abstract representation of that system. Architecture is concerned with making sure the software system will meet the requirements of the product, as well as ensuring that future requirements can be addressed. The architecture step also addresses interfaces between the software system and other software products, as well as the underlying hardware or the host operating system.

Implementation, testing and documenting

Implementation is the part of the process where software engineers actually program the code for the project.

Software testing is an integral and important part of the software development process. This part of the process ensures that bugs are recognized as early as possible.

Documenting the internal design of software for the purpose of future maintenance and enhancement is done throughout development. This may also include the authoring of an API, be it external or internal.

Deployment and maintenance

Deployment starts after the code is appropriately tested, is approved for release and sold or otherwise distributed into a production environment.

Software Training and Support is important because a large percentage of software projects fail because the developers fail to realize that it doesn't matter how much time and planning a development team puts into creating software if nobody in an organization ends up using it. People are often resistant to change and avoid venturing into an unfamiliar area, so as a part of the deployment phase, it is very important to have training classes for new clients of your software.

Maintenance and enhancing software to cope with newly discovered problems or new requirements can take far more time than the initial development of the software. It may be necessary to add code that does not fit the original design to correct an unforeseen problem or it may be that a customer is requesting more functionality and code can be added to accommodate their requests. It is during this phase that customer calls come in and you see whether your testing was extensive enough to uncover the problems before customers do. If the labor cost of the maintenance phase exceeds 25% of the prior-phases' labor cost, then it is likely that the overall quality, of at least one prior phase, is poor. In that case, management should consider the option of rebuilding the system (or portions) before maintenance cost is out of control.

Bug Tracking System tools are often deployed at this stage of the process to allow development teams to interface with customer/field teams testing the software to identify any real or perceived issues. These software tools, both open source and commercially licensed, provide a customizable process to acquire, review, acknowledge, and respond to reported issues

2. Explain flowchart in detail? (JAN 2011)

Flowcharts

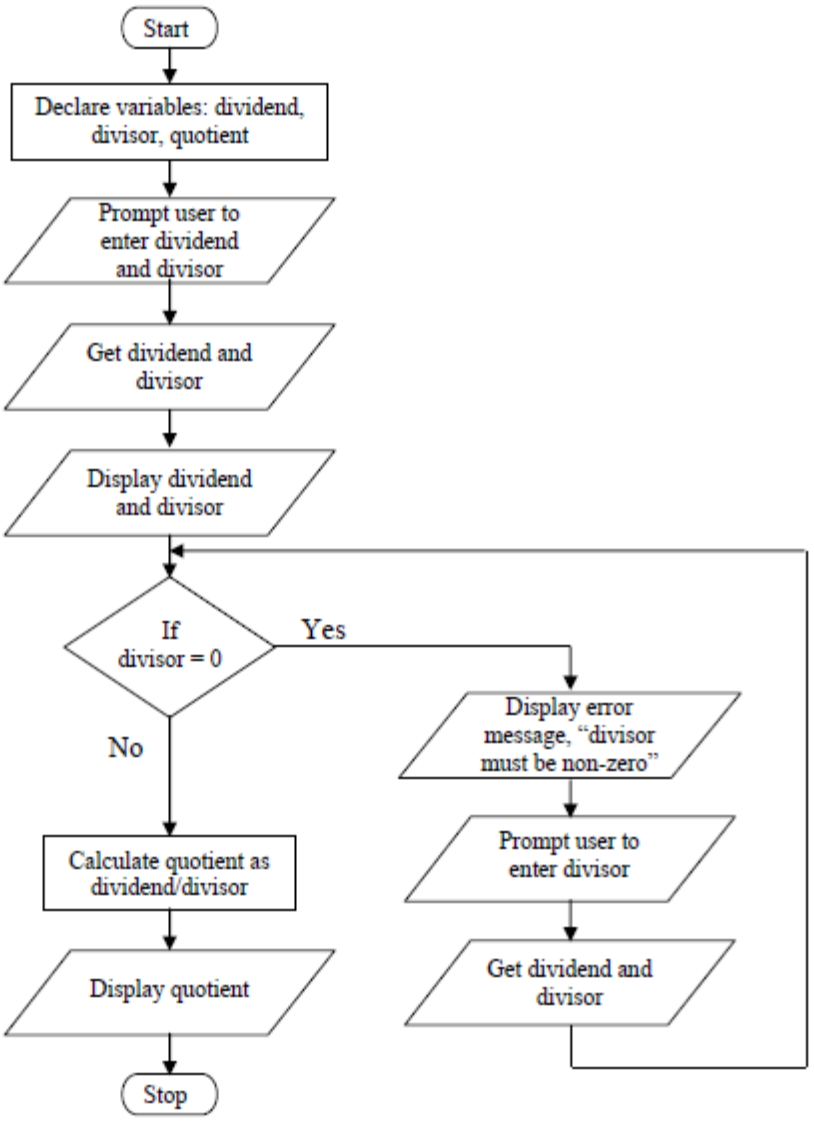
- A graphical tool that *diagrammatically* depicts the steps and structure of an algorithm or program
- Symbols[6,7] (the most commonly used ones)

Symbol	Name/Meaning	Symbol	Meaning
	Process – Any type of internal operation: data transformation, data movement, logic operation, etc.		Connector – connects sections of the flowchart, so that the diagram can maintain a smooth, linear flow
	Input/Output – input or output of data		Terminal – indicates start or end of the program or algorithm
	Decision – evaluates a condition or statement and branches depending on whether the evaluation is true or false		Flow lines – *arrows* that indicate the direction of the progression of the program

- General rules for flowcharts
  - All symbols of the flowchart are connected by flow lines (note *arrows*, not lines)
  - Flowlines enter the top of the symbol and exit out the bottom, except for the Decision symbol, which can have flow lines exiting from the bottom or the sides
  - Flowcharts are drawn so flow generally goes from top to bottom
  - The beginning and the end of the flowchart is indicated using the Terminal symbol

Flowchart Example

Express an algorithm to get two numbers from the user (dividend and divisor), testing to make sure that the divisor number is not zero, and displaying their quotient using a *flowchart*.



3. Write an algorithm to compute the factorial of a number n. (JAN 2012)

1.Take a number as a input from the user.
2. Initialize a variable fact=1

3. fact=fact*number
4. number=number-1
5. repeat step 3 to 5 if number>0
6. print the value of fact variable

4. Describe in detail about algorithm? Give example

'Algorithm' is a really a *recipe*, that is, a set of step-by-step instructions that takes raw ingredients and produces a tasty result. In general, an algorithm can be described as a procedure to solve a problem.

In the context of computer programming, an *algorithm*, is defined as a: "well-ordered collection of unambiguous and effectively computable operations, that when executed, produces a result and halts in a finite amount of time."[1]

Characteristics of an Algorithm

- Well-ordered: the steps are in a clear order
- Unambiguous: the operations described are understood by a computing agent without further simplification
- Effectively computable: the computing agent can actually carry out the operation

Method for Developing an Algorithm

1. Define the problem: State the problem you are trying to solve in *clear* and *concise* terms.
2. List the *inputs* (information needed to solve the problem) and the *outputs* (what the algorithm will produce as a result)
3. Describe the steps needed to convert or manipulate the inputs to produce the outputs. Start at a high level first, and keep refining the steps until they are *effectively computable* operations.
4. Test the algorithm: choose data sets and verify that your algorithm works!

5. Elaborate pseudo code with example.

Pseudo-Code is simply a numbered list of instructions to perform some task. In this course we will enforce three standards for good pseudo code

1. Number each instruction. This is to enforce the notion of an *ordered sequence of ... operations*. Furthermore we introduce a *dot* notation (e.g. 3.1 come *after* 3 but *before* 4) to number subordinate operations for conditional and iterative operations
2. Each instruction should be *unambiguous* (that is the computing agent, in this case the reader, is capable of carrying out the instruction) and *effectively computable* (do-able).
3. Completeness. Nothing is left out.

Pseudo-code is best understood by looking at examples. Each example below demonstrates one of the control structures used in algorithms : sequential operations, conditional operations, and iterative operations. We also list all variables used at the end of the pseudo-code.

Example #1 - Computing Sales Tax : Pseudo-code the task of computing the final price of an item after figuring in sales tax. Note the three types of instructions: input (get), process/calculate (=) and output (display)

1. get price of item
2. get sales tax rate
3. sales tax = price of time times sales tax rate
4 final prince = price of item plus sales tax
5. display final price
6. halt

Pseudocode Language Constructs[5]

- Computation/Assignment
  - **Compute** var1 as the sum of x and y
  - **Assign** expression to var2
  - **Increment** counter1
- Input/Output
  - Input: **Get** var1, var2, ...
  - Output: **Display** var1, var2, ...
- Selection

  Single-Selection IF

  1. **IF** *condition* **THEN** (IF condition is true, then do subordinate statement 1, etc. If condition is false, then skip statements)
     1.1 statement 1
     1.2 etc.

  Double-Selection IF

  2. **IF** *condition* **THEN** (IF condition is true, then do subordinate statement 1, etc. If condition is false, then skip statements and execute statements under ELSE)
     2.1 statement 1
     2.2 etc.
  3. **ELSE** (else if condition is not true, then do subordinate statement 2, etc.)
     3.1 statement 2
     3.2 statement 3
  4. **SWITCH** *expression* TO
     4.1 case 1: action1
     4.2 case 2: action2
     4.3 etc.
     4.4 default: actionx
- Repetition

  5. **WHILE** *condition* (while condition is true, then do subordinate statements)
     5.1 statement 1
     5.2 etc.

  DO – WHILE structure (like WHILE, but tests condition at the *end* of the loop. Thus, statements in the structure will always be executed at least once.)

  6. **DO**
     6.1 statement 1
     6.2 etc.
  7. **WHILE** *condition*

Pseudocode Language Constructs[5]

- Computation/Assignment
  - **Compute** var1 as the sum of x and y
  - **Assign** expression to var2
  - **Increment** counter1
- Input/Output
  - Input: **Get** var1, var2, ...
  - Output: **Display** var1, var2, ...
- Selection

  Single-Selection IF

  1. **IF** *condition* **THEN** (IF condition is true, then do subordinate statement 1, etc. If condition is false, then skip statements)
     1.1 statement 1
     1.2 etc.

  Double-Selection IF

  2. **IF** *condition* **THEN** (IF condition is true, then do subordinate statement 1, etc. If condition is false, then skip statements and execute statements under ELSE)
     2.1 statement 1
     2.2 etc.
  3. **ELSE** (else if condition is not true, then do subordinate statement 2, etc.)
     3.1 statement 2
     3.2 statement 3
  4. **SWITCH** *expression* TO
     4.1 case 1: action1
     4.2 case 2: action2
     4.3 etc.
     4.4 default: actionx
- Repetition
  5. **WHILE** *condition* (while condition is true, then do subordinate statements)
     5.1 statement 1
     5.2 etc.

  DO – WHILE structure (like WHILE, but tests condition at the *end* of the loop. Thus, statements in the structure will always be executed at least once.)

  6. **DO**
     6.1 statement 1
     6.2 etc.
  7. **WHILE** *condition*

FOR structure (a specialized version of WHILE for repeating execution of statements a specific number of times)

8. **FOR** *bounds on repetition*
    8.1 statement 1
    8.2 etc.

Pseudocode Example

Express an algorithm to get two numbers from the user (dividend and divisor), testing to make sure that the divisor number is not zero, and displaying their quotient using *pseudocode*

1. Declare variables: dividend, divisor, quotient
2. Prompt user to enter dividend and divisor
3. Get dividend and divisor
4. IF divisor is equal to zero, THEN
    4.1. DO
        4.1.1. Display error message, "divisor must be non-zero"
        4.1.2. Prompt user to enter divisor
        4.1.3. Get divisor
    4.2. WHILE divisor is equal to zero
5. ENDIF
6. Display dividend and divisor
7. Calculate quotient as dividend/divisor
8. Display quotient

6. Explain sequence logic, selection logic, and iteration logic design structure in pseudocode. (JAN 2011)

Pseudocode is a tool for planning, defining, or documenting the contents of a program routine or module. As the name implies, pseudocode is similar to (and often based on) real code.

Pseudocode is an excellent tool for planning or designing program logic and computational algorithms. Because it resembles real code, programmers find it easy to use and to understand. Pseudocode is not a good tool for describing control structures, particularly when several nested decisions are involved. Because pseudocode resembles a programming language, it is not well suited to planning or designing manual procedures.

In part, because pseudocode so closely resembles real code, some designers tend to write (rather than plan or design) the code. Writing the code twice (once in pseudocode and once in real code) is a waste of time. When the designer worries about coding details, crucial design considerations can easily be overlooked. Also, programmers resent such over specification.

Even if the pseudocode is well done, programmers sometimes resent it. Specifying algorithms in what is essentially a high-level language limits the programmer's flexibility. Additionally, the programmer can fail to distinguish between the analyst's coding technique and the analyst's design, and the result may be criticism (even rejection) of a perfectly good design based on inappropriate criteria.

Sequence

Perhaps the easiest way to define sequential logic is by coding an algebra-like expression, for example,

```
COUNT = 0
```

or

```
STOCK = STOCK + QUANTITY
```

Such details as the sequence of operations and the rules for using parentheses should be consistent with the language to be used for writing the actual code. Data names should be taken from the data dictionary.

Input and output instructions are explicitly defined in pseudocode; for example,

```
READ data FROM source
```

and

```
WRITE data TO destination
```

where *data* is a list of variables, a data structure, or a record, and *source* and *destination* refer to a file or a database.

Blocks of logic

A block can contain any set of sequence, decision (selection), and/or repetition (iteration) logic. Once defined, the block's instructions can be referenced by coding a PERFORM instruction:

```
PERFORM block
```

For example, the instructions

```
COUNT = 0
ACCUMULATOR = 0
```

might be assigned the block name INITIALIZE. Subsequently, the instruction

```
PERFORM INITIALIZE
```

executes all the instructions in the block.

To distinguish between formal subroutines and internal procedures, some analysts use:

```
PERFORM block USING list
```

for a subroutine, where *list* designates a list of variables passed between the calling routine and the subroutine.

Decision or selection

The general form of a decision (selection) block is:

```
IF condition
    THEN
       PERFORM block-1
   ELSE
       PERFORM block-2
ENDIF
```

For example,

```
IF HOURS > 40
    THEN
      PERFORM OVERTIME
ELSE
    PERFORM REGULAR
ENDIF
```

By convention, the THEN block is executed if the condition is true and the ELSE block is executed if the condition is false.

Note the ENDIF. A feature of most pseudocodes is that each block of logic is clearly delimited. A decision block always begins with IF and ends with ENDIF, so there is no ambiguity. Note also the use of indentation. It makes the block easy to read.

It is possible to nest decision logic. For example,

```
IF condition-1
    THEN
       IF condition-2
      THEN
        PERFORM block-a
     ELSE
        PERFORM block-b
    ENDIF
 ELSE
    PERFORM block-c
ENDIF
```

Note how indentation highlights the relationship among these instructions. Note also how IF and ENDIF clearly delimit both decision blocks.

Repetition or iteration

DO WHILE logic tests for the terminal condition at the top of the loop. For example,

```
WHILE condition DO

   PERFORM block

ENDWHILE
```

Note the use of indentation and the way WHILE and ENDWHILE delimit the block. The REPEAT UNTIL structure tests for the terminal condition at the bottom of the loop:

```
REPEAT

   PERFORM block

UNTIL condition
```

Some analysts use a pseudocode structure much like a DO loop to define a count-controlled loop:

```
DO index = initial TO limit

   PERFORM block

ENDDO
```

Again, note the indentation and the ENDDO.

The case structure

A common programming problem involves selecting among several alternative paths. Although nested decision statements could be used to define such logic, the case structure is a better option when nesting goes beyond three or four levels.

The general form of a case structure is:

```
SELECT variable

   CASE (value-1) block-1
   CASE (value-2) block-2
   .
   .
   .
   DEFAULT CASE block-n

 ENDSELECT
```

The option selected by the case structure depends on the value of the control variable following the keyword SELECT. If the variable contains *value-1*, then *block-1* is executed; if it contains *value-2*, then *block-2* is executed, and so on. The DEFAULT CASE is coded to define the logic to be executed if the control variable contains none of the listed values. A case structure is delimited by ENDSELECT. Once again, indentation makes the structure easy to read.

For example, the following pseudocode routine accepts a code (the TRANSACTION TYPE) and, based on the value of the code, passes control to one of three lower-level routines.

```
SELECT TRANSACTION TYPE

   CASE (modify)     PERFORM MODIFY STOCK
   CASE (add)        PERFORM ADD RECORD
   CASE (delete)     PERFORM DELETE RECORD
   DEFAULT CASE      PERFORM TRANSACTION ERROR
ENDSELECT
```

7. Explain in detail about the word processing package. (JAN 2012)
   Features of MS-Word:
   - Adjustment of Font size, type, colour, Subscript, Superscript, Spacing, Text Effects etc,.
   - Bullets and Numberings
   - Changing case
   - Borders and Shadings etc,.
   - Find and Replace
   - Alignment (left, right, centre)
   - Spell check
   - Insertion of tables, charts, pictures, objects, clipart, shapes.
   - Mail merge

7. Describe about spreadsheet packages. (JAN 2012)
   Features of Spread sheets:
   - Functions and Arithmetic Formulas
   - Auto calculation
   - Charts
   - Database
   - Import and Export objects / files.

- Filters.
- Conversion formulas.
- Alignment of text inside a cell

8. Write the pseudocode to add two matrices. (JAN 2012)

Matrix Addition Pseudo code

```
INIT add
FOR each x from 0 to 15
FOR each y from 0 to 15
SET input1[x][y]=NULL
SET input2[x][y]=NULL
SET tot[x][y]=NULL
INCREMENT y
END FOR
INCREMENT x
END FOR
SET r=0, c=0
READ c and r
FOR each x from 0 to less than r
FOR each y from 0 to less than c
READ input1[x][y]
INCREMENT y
END FOR
INCREMENT x
END FOR
FOR each x from 0 to less than r
FOR each y from 0 to less than c
READ input2[x][y]
INCREMENT y
END FOR
INCREMENT x
END FOR
FOR each x from 0 to less than r
FOR each y from 0 to less than c
COMPUTE tot[x][y]=input1[x][y]+input2[x][y]
PRINT tot[x][y]
INCREMENT y
END FOR
INCREMENT x
END FOR
END
```

9. Discuss about the graphics package along with its various features in detail.
Features of graphics packages:
   - Fonts
   - Bullets and Numbering
   - Alignment
   - Line spacing
   - Change case
   - Replace fonts
   - Slide Design
   - Slide Layout

10. Write the algorithm and pseudocode to check the given no is prime or not

ALGORITHM

1.Start
2.Read the value of n
3.for i=1to n
4.check if n%i==0 then increment the value of c by 1
5.Check if c==2 then print number is prime
6.else print the number is not prime

PSEUDOCODE

```
BEGIN
READ n
For i=1 to n DO
IF n%i==0 THEN
INCREATMENT c by 1
END IF
END FOR
IF c==2
PRINT Prime Number
ELSE
PRINT not a Prime Number
END
```

11. Briefly explain about the personal assistant package.
    - Address book
    - Calendar
    - To-do List
    - Inventory Books
    - Investments Book

UNIT – IV
PART – A

1. What are the different data types available in 'C'?
There are four basic data types available in 'C'.
1. int
2. float
3. char
4. double

2. What are Keywords?
Keywords are certain reserved words that have standard and pre-defined meaning in 'C'. These keywords can be used only for their intended purpose.

3. What is an Operator and Operand?
An operator is a symbol that specifies an operation to be performed on operands.
Example: *, +, -, / are called arithmetic operators.The data items that operators act upon are called operands.**Example:** a+b; In this statement a and b are called operands.

4. What is Ternary operators or Conditional operators?
Ternary operators is a conditional operator with symbols ? and :
Syntax: variable = exp1 ? exp2 : exp3

If the exp1 is true variable takes value of exp2. If the exp2 is false, variable takes the value of exp3.

5. What are the Bitwise operators available in 'C'?

& - Bitwise AND, | - Bitwise OR, ~ - One's Complement, >> - Right shift, << - Left shift, ^ - Bitwise XOR are called bit field operators.

Example: k=~j; where ~ take one's complement of j and the result is stored in k.

6. What are the logical operators available in 'C'?

The logical operators available in 'C' are

&& - Logical AND, || - Logical OR, ! - Logical NOT

7. What is the difference between Logical AND and Bitwise AND?

Logical AND (&&): Only used in conjunction with two expressions, to test more than one condition. If both the conditions are true the returns 1. If false then return 0.

AND (&): Only used in Bitwise manipulation. It is a unary operator.

8. What is the difference between '=' and '==' operator?

Where = is an assignment operator and == is a relational operator.

Example:while (i=5) is an infinite loop because it is a non zero value and while (i==5) is true only when i=5.

9. What is type casting?

Type casting is the process of converting the value of an expression to a particular data type.**Example: int x,y;** c = (float) x/y; where a and y are defined as integers. Then the result of x/y is converted into float.

10. What is conversion specification?

The conversion specifications are used to accept or display the data using the INPUT/OUTPUT statements.

11. What is the difference between 'a' and "a"?

'a' is a character constant and "a" is a string.

12. What is the difference between if and while statement?

If while

(i) It is a conditional statement (i) It is a loop control statement(ii) If the condition is true, it executes some statements.(ii) Executes the statements within the while block if the condition is true. (iii) If the condition is false then it stops the execution the statements. (iii) If the condition is false the control is transferred to the next statement of the loop.

13. What is the difference between while loop and do…while loop?

In the while loop the condition is first executed. If the condition is true then it executes the body of the loop. When the condition is false it comes of the loop. In the do…while loop first the statement is executed and then the condition is checked. The do…while loop will execute at least one time even though the condition is false at the very first time.

14. What is a Modulo Operator?

'%' is modulo operator. It gives the remainder of an integer division

Example:a=17, b=6. Then c=%b gives 5.

15. How many bytes are occupied by the int, char, float, long int and double?

int - 2 Bytes, char - 1 Byte, float - 4 Bytes, long int - 4 Bytes, double - 8 Bytes

16. What are the types of I/O statements available in 'C'?

There are two types of I/O statements available in 'C'.

Formatted I/O Statements, Unformatted I/O Statements

17. What is the difference between ++a and a++?

++a means do the increment before the operation (pre increment)

a++ means do the increment after the operation (post increment)

Example:

```
a=5;
x=a++; /* assign x=5*/
y=a; /*now y assigns y=6*/
x=++a; /*assigns x=7*/
```

18. What is a String?

String is an array of characters.

19. What is a global variable?

The global variable is a variable that is declared outside of all the functions. The global variable is stored in memory, the default value is zero. Scope of this variable is available in all the functions. Life as long as the program's execution doesn't come to an end.

20. What are the Escape Sequences present in 'C'

\n - New Line, \b – Backspace, \t - Form feed, \' - Single quote,,\\ - Backspace, \t – Tab, \r - Carriage return, \a – Alert, \" - Double quotes

21. Construct an infinite loop using while?

```
while (1)
{}
```

Here 1 is a non zero, value so the condition is always true. So it is an infinite loop.

22. What will happen when you access the array more than its dimension?

When you access the array more than its dimensions some garbage value is stored in the array.

23. Write the limitations of getchar( ) and sacnf( ) functions for reading strings

getchar( )

To read a single character from stdin, then getchar() is the appropriate.

scanf( )

scanf( ) allows to read more than just a single character at a time.

24. What is the difference between scanf() and gets() function?

In scanf() when there is a blank was typed, the scanf() assumes that it is an end. gets() assumes the enter key as end. That is gets() gets a new line (\n) terminated string of characters from the keyboard and replaces the '\n' with '\0'.

25. What is a Structure?

Structure is a group name in which dissimilar data's are grouped together.

26. What is meant by Control String in Input/Output Statements?

Control Statements contains the format code characters, specifies the type of data that the user accessed within the Input/Output statements.

27. What is Union?

Union is a group name used to define dissimilar data types. The union occupies only the maximum byte of the data type. If you declare integer and character, then the union occupies only 2 bytes, whereas structure occupies only 3 bytes.

28. What is the output of the programs given below?

```
main()
 {
 float a;
 int x=6, y=4;
 a=(float) x\y;
 printf("Value of a=%f",a);
 }
```

Output:

1.500000

29. Declare the Structure with an example?

```
struct name
{c
har name[10];
int age;
float salary;
} e1, e2;
```

30. Declare the Union with an example?

```
union name
{
char name[10];
int age;
float salary;
    } e1, e2;
```

31. What are the main features and applications of C language? (JAN 2011)

C programming language that was capable of both high level, machine independent programming and would still allow the programmer to control the behavior of individual bits of information. C includes bitwise operators along with powerful pointer manipulation capabilities.

32. Define with example integer and floating type of data in C language. (JAN 2011)

The int is used to define integer numbers. An integer occupies 2 bytes memory space and its value range limited to -32768 to +32767 (that is, -215 to +215-1).

```
int count
count=5
```

The float is used to define floating point numbers. The float data type is used to store fractional numbers (real numbers) with 6 digits of precision. Floating point numbers are denoted by the keyword float.

```
float miles
miles = 4.5
```

33. What are the operators exclusively used with pointers?

The address operator, &

PART – B

1.Explain in detail about 'C' declarations and variables. (JAN 2011)

Variables:

- It is an identifier used to represent specified type of information.
- A variable may take different values at different times.

Syntax:

data_type v1,v2,..;
Eg:
int a,b;

Rules for Naming Variable:

1. A variable name can be any combination of 1 to 8 alphabets or digits or underscore.
2. The first character must be an alphabet.
3. The Length of the variable cannot exceed up to 8 characters long.
4. No commas or Blank space are allowed.
5. No special symbol can be used in a variable name but underscore can be used.

Scope of variable

• Two scope types

1. Local Variables.
2. Global Variables.

Local Variables:

```
main()
{
int a=5;
}
```

Global Variables:

```
int a=5;
```

```
main()
{
printf(“the values is %d”,a);
}
```

2. Explain in detail about the constants, expressions and statements in ‘C’.

Constants:

The item whose values cannot be changed during the execution of program.

C Constants

- **Numeric constant :**
  - Integer constants
  - Real constant
- **Character constant :**
  - Single Character constant
  - string constant.

String Constants :

A string constant is a sequence of characters enclosed in double quotes, the characters may be letters, numbers, special characters and blank spaces etc. At the end of string '\0' is automatically placed.

Example :

"HI“, "Prabhu“

Expressions:

An expression represents a single data item, such as number or a character. Logical conditions that are true or false are represented by expressions.

Example: a = p – q / 3 + r * 2 – 1

Statements

- Assignment Statements – Definition and examples
- Null Statements – Definition and examples
- Block of statements – Definition and examples
- Expression statements – Definition and examples
- Declaration statements – Definition and examples

Assignment statement:

The assignment operator used for assigning values to the variables.

Example : basic = 2977;dob = 2977;

Block of statement:

Block contains several statements that are enclosed within a pair of braces {}. These can be any of expression, assignments and keywords etc.

Example :

```
int a=2977;

float b=29.77;

printf(%22%25d%25f%22, a,b);
```

Expression Statement:

These consists of expressions and can be arithmetic, relational or logical.

Example :

```
a=29

b=a+77;

fun(a,b);
```

3.Discuss about the various data types in ‘C’.

DATATYPES:

Data type is the type of the data, that are going to access within the program. 'C' supports different data types, each data type may have predefined memory requirement and storage representation.

'C' supports the following 4 classes of data types

'C' Data Types

Primary	**Userdefined**	**Derived**	**Empty**
Char		arrays	
Int	typedef	pointer	void
Float	enum	structures	
Double		union	

The four basic data types are

a. Integer

- These are whole numbers, both positive and negative. Unsigned integers (positive values only) are supported. In addition, there are short and long integers.

- The keyword used to define integers is,int
- An example of an integer value is 32.
- An example of declaring an integer variable called **sum** is,
  int sum;
  sum = 20;

b. Floating Point

- These are numbers which contain fractional parts, both positive and negative.
- The keyword used to define float variables is,float
- An example of a float value is 34.12.
- An example of declaring a float variable called **money** is,
  float money;
  money = 0.12;

c. Double

- These are exponentional numbers, both positive and negative. The keyword used
- to define double variables is,double
- An example of a double value is 3.0E2.
- An example of declaring a double variable called **big** is,
  double big;
  big = 312E+7;

d. Character

- These are single characters.
- The keyword used to define character variables is,char
- An example of a character value is the letter **A**.
- An example of declaring a character variable called **letter** is,
  char letter;
  letter = 'A';

Note the assignment of the character A to the variable letter is done by enclosing the value in **single quotes**.

User defined variables

i. typedef()
ii. enum()

i. Type declaration:

syntax:

typedef datatype identifier;

Eg: typedef int marks;

marks m1,m2;

ii. Enumerated data type:

syntax:

enum identifier {val1,val2,…}

eg: enum day(mon,tues,wed}

4.Describe the various types of operators in 'C' language along with its priority.

1.Assignment operators

Assignment operators are used to assign a value or an expression or a value of a variable to another variable.

Syntax :

variable=expression (or) value ;

Example :

x=10;

2. Arithmatic Operators:

The symbols of the arithmetic operators are:-

Operation	Operator	Comment	Value of Sum before	Value of sum after
Multiply	*	sum = sum * 2;	4	8
Divide	/	sum = sum / 2;	4	2
Addition	+	sum = sum + 2;	4	6
Subtraction	-	sum = sum -2;	4	2
Increment	++	++sum;	4	5
Decrement	--	--sum;	4	3
Modulus	%	sum = sum % 3;	4	1

Example:

```
#include <stdio.h>
main()
{ int sum =50;
float modulus;
modulus = sum % 10;
printf("The %% of %d by 10 is %f\n", sum, modulus);
```

3. Increment & Decrement Operator:

'C' has two way useful operators not generally found in other languages, these are the
increment (++) and decrement (--) operators.'+' & '-' operators are called unary operators.
Because they acts upon only one variable.

++x - Pre increment
--x - Pre decrement
x++ - Post increment
x-- - Post decrement

4. Relational Operator:

Relational operators are used to compare two or more operands. Operands may be variables, constants or expression.

Operator	Meaning
<	is less than
<=	is less than or equal to
>	is greater than
>=	is greater than or equal to
==	is equal to
!=	is not equal to

5.Logical operators :

Logical operators are used to combine the results of two or more conditions.

Operator	Meaning	Example	Return value
• &&	Logical AND	(9>2)&&(17>2)	1
• ||	Logical OR	(9>2)||(17==7)	1
• !	Logical NOT	29!=29	0

6.Conditional Operator

• Checks the condition and executes the statement.

Ternary Operator (condition ? Exp1:Exp2)

Eg:

```
void main()
{
int a=5,b=2,big;
big=(a>b)?a:b;
printf("Largest number is %d",big);
}
```

Output:

Largest number is a

7. BitWise Operator:

- Bitwise operators are used to manipulate the data at bit level.
- It operates on integers only

Operator	**Meaning**

&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<	Shift left
>>	Shift right
~	One's complement

8.Special Operator:

- ,
- sizeof
- & and *
- . and →

5.Explain about the various decision making statements in 'C' language. (JAN 2011)

if statement:

• The if statement is a decision making statement. It is used to control the flow of execution
of the statements and also used to test logically whether the condition is true or false. It is
always used in conjunction with condition. This statement is used when a question requires
answer.

• If the condition is true, then the True statements are executed.

• 'True statements' may be a single statement or group of statements. If the condition is false
Then the true statements are not executed, instead the program skip past it. The condition is
given by the relational operator like ==, !=, <=, >=, etc.

Example:

```
#include <stdio.h>
main()
{
int number;
int valid = 0;
while( valid == 0 )
{
printf("Enter a number between 1 and 10 □");
scanf("%d", &number);
```

```
/* assume number is valid */
valid = 1;
if( number < 1 )
{
printf(“Number is below 1. Please re-enter\n”);
valid = 0;
}
 if( number >10 )
{
printf(“Number is above 10. Please re-enter\n”);
valid = 0;
}
}
printf(“The number is %d\n”, number );
}
```

If-else statement:

It is basically two way decision making statement and always used in conjunction with condition. It is used to control the flow of execution and also used to carry out the logical test and then pickup one of the two possible actions depending on the logical test.

Program to display the types of character using if-else satement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char chr;
clrscr();
printf("Enter a single character : ");
scanf(",&chr);
if((chr >= 'a' && chr <= 'z') || (chr >= 'A' && chr <= 'Z'))
printf("Entered character is an alphabetic.\n");
else
if(chr >= '0' && chr <= '9‘)
printf("Entered character is an digit.\n");
else
printf("Entered character is an special character.\n");
getch();
}
```

Nested if:

When a series of if..else statements are occurred in a program, we can write an entire
if..else statement in another if..else statement called nesting, and the statement is called nested if.

Example:

```
#include <stdio.h>
main()
{
 int invalid_operator =0;
```

```
char operator;
float number1, number2, result;
printf("Enter two numbers and an operator in the format\n");
printf(" number1 operator number2\n");
scanf("%f %c %f", &number1, &operator, &number2);
if(operator == '*')
result = number1 * number2;
else if(operator == '/')
result = number1 / number2;
else if(operator == '+')
result = number1 + number2;
else if(operator == '-')
result = number1 – number2;
else
invalid_operator = 1;
if( invalid_operator != 1 )
printf("%f %c %f is %f\n", number1, operator, number2, result );
else
printf("Invalid operator.\n");
}
```

6.Write short notes on the following: (JAN 2012)

'for' loop
'while' loop
'dowhile' loop
'Switch case '

1. Iteration, For Loops

The basic format of the for statement is,

```
for( start condition; continue condition; re-evaulation )
program statement;
```

Example:

```
/* sample program using a for statement */
#include <stdio.h>
main() /* Program introduces the for statement, counts to ten */
{ int count;
for( count = 1; count <= 10; count = count + 1 )
printf("%d ", count );
printf("\n");
}
```

The program declares an integer variable count. The first part of the for statement **for (count = 1; initialized the value of count to 1.**

The for loop continues with the condition count <= 10; evaluates as TRUE. As the variable count has just been initialized to 1, this condition is TRUE and so the program statement printf("%d ", count); is executed, which prints the value of count to the screen, followed by a space character.

Next, the remaining statement of the for is executed count = count + 1); which adds one to the current value of count. Control now passes back to the conditional test, count <= 10; which evaluates as true, so the program statement printf("%d ", count); is executed.

Count is incremented again, the condition re-evaluated etc, until count reaches a value of 11.

When this occurs, the conditional test count <= 10; evaluates as FALSE, and the for loop terminates, and program control passes to the statement printf("\n"); which prints a newline, and then the program terminates, as there are no more statements left to execute.

2. The While Statement

The while provides a mechanism for repeating C statements whilst a condition is true. Its format is, while(condition) program statement;Somewhere within the body of the while loop a statement must alter the value of the condition to allow the loop to finish.

Example:

/* Sample program including while */

```
#include <stdio.h>
main()
{ int loop = 0;
while( loop <= 10 )
{
printf("%d\n", loop);
++loop;
}
}
```

The above program uses a while loop to repeat the statements
printf("%d\n",loop);
++loop;
the value of the variable loop is less than or equal to 10.

3. The Do While Statement

The do { } while statement allows a loop to continue whilst a condition evaluates as
TRUE (non-zero). The loop is executed as least once.

Example:

/* Demonstration of DO...WHILE */

```
#include <stdio.h>
main()
{ int value, r_digit;
printf("Enter the number to be reversed.\n");
scanf("%d", &value);
do {
r_digit = value % 10;
printf("%d", r_digit);
value = value / 10;
} while( value != 0 );
printf("\n");
}
```

The above program reverses a number that is entered by the user. It does this by using the modulus % operator to extract the right most digit into the variable r_digit. The original number is then divided by 10, and the operation repeated whilst the number is

not equal to 0.

4. **Switch Case:**

The switch case statement is a better way of writing a program when a series of if elses occurs.

The general format is,

```
switch ( expression ) {
case value1:
program statement;
program statement;
.....
break;
case valuen:
program statement;
.....
break;
default:
.....
.....
break;
}
```

The keyword break must be included at the end of each case statement. The default clause is optional, and is executed if the cases are not met. The right brace at the end signifies the end of the case selections.

Example:

```
#include <stdio.h>
main()
{
 int menu, numb1, numb2, total;
printf("enter in two numbers □");
scanf("%d %d", &numb1, &numb2 );
printf("enter in choice\n");
printf("1=addition\n");
printf("2=subtraction\n");
scanf("%d", &menu );
switch( menu )
{
case 1: total = numb1 + numb2; break;
case 2: total = numb1 – numb2; break;
default: printf("Invalid option selected\n");
}
 if( menu == 1
)
printf("%d plus %d is %d\n", numb1, numb2, total );
else if( menu == 2 )
printf("%d minus %d is %d\n", numb1, numb2, total );
}
```

The above program uses a switch statement to validate and select upon the users input choice, simulating a simple menu of choices.

7. Explain briefly about the formatted and unformatted I/O function in 'C'. (JAN 2012) (JAN 2011)

1 printf ():

printf() is actually a function (procedure) in C that is used for printing variables and text. Where text appears in double quotes "", it is printed without modification. There are some exceptions however.This has to do with the \ and % characters. These characters are modifiers, and for the present the \ followed by the n character represents a newline character.

Example:

```
#include <stdio.h>
main()
{
printf("Programming in C is easy.\n");
printf("And so is Pascal.\n");
}
```

@ Programming in C is easy.
And so is Pascal.

FORMATTERS for printf are,

Cursor Control Formatters

\n newline
\t tab
\r carriage return
\f form feed
\v vertical tab

2. Scanf ():

Scanf () is a function in C which allows the programmer to accept input from a keyboard.

Example:

```
#include <stdio.h>
main() /* program which introduces keyboard input */
{ int number;
printf("Type in a number \n");
scanf("%d", &number);
printf("The number you typed was %d\n", number);
}
```

Formatters For scanf()

The following characters, after the % character, in a scanf argument, have the following effect.

D read a decimal integer
o read an octal value
x read a hexadecimal value
h read a short integer

l read a long integer
f read a float value
e read a double value
c read a single character
s read a sequence of characters
[...] Read a character string. The characters inside the brackets

3. Accepting Single Characters From The Keyboard

Getchar, Putchar

getchar() gets a single character from the keyboard, and putchar() writes a single character from the keyboard.

Example:

The following program illustrates this,

```
#include <stdio.h>
main()
{ int i;
int ch;
for( i = 1; i<= 5; ++i ) {
ch = getchar();
putchar(ch);
}}
```

The program reads five characters (one for each iteration of the for loop) from the keyboard. Note that getchar() gets a single character from the keyboard, and putchar() writes a single character (in this case, ch) to the console screen.

8. (a) Describe in detail about type conversions in 'C' with example.

Implicit Type Conversion

For arithmetic/relational operators, lower variable type is promoted to higher variable type

- Among common variable types, the type order is below.
- double (highest)
- float
- Int
- Char(lowest)

For assignments, value of right side is converted to type of left
The value in code snippets like below depend on the rules

```
double d = 12.56;
int i = d;
i = 34.56;
```

Value of i is first 12, and then 34

Explicit Type conversion of variables

Explicit type casting of variables can be done by

```
double d = -12.56;
```

int i = (int) d;

- Value of i gets truncated to -12
- Conversion from higher type to lower type may lose information
- Conversion from lower type to higher type should not lose information
- Avoid implicit type conversion as much as possible.

(b) Define delimiters. List them. Give an example program using various delimiters.

A delimiter is a sequence of one or more characters used to specify the boundary between separate, independent regions in plain text or other data streams. An example of a delimiter is the comma character, which acts as a field delimiter in a sequence of comma-separated values.

Delimiters	Description
(and)	Parentheses. The Lisp programming language syntax is cited as recognizable primarily by its use of parentheses.[7]
{ and }	Curly brackets.[8]
< and >	Angle brackets.[9]
" and "	commonly used to denote string literals.[10]
' and '	commonly used to denote string literals.[10]
<? and ?>	used to indicate XML processing instructions.[11]
/* and */	used to denote comments in some programming languages.[12]
<% and %>	used in some web templates to specify language boundaries. These are also called template delimiters.[13]

9. Explain the following: (JAN 2011)

1. if..else statement
2. nested if…else
3. switch statement

If-else statement:

It is basically two way decision making statement and always used in conjunction with condition. It is used to control the flow of execution and also used to carry out the logical test and then pickup one of the two possible actions depending on the logical test.

Program to display the types of character using if-else satement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char chr;
clrscr();
printf("Enter a single character : ");
scanf("",&chr);
if((chr >= 'a' && chr <= 'z') || (chr >= 'A' && chr <= 'Z'))
```

```
printf("Entered character is an alphabetic.\n");
else
if(chr >= '0' && chr <= '9‘)
printf("Entered character is an digit.\n");
else
printf("Entered character is an special character.\n");
getch();
}
```

Nested if:

When a series of if..else statements are occurred in a program, we can write an entire
if..else statement in another if..else statement called nesting, and the statement is called nested if.

Example:

```
#include <stdio.h>
main()
{
 int invalid_operator =0;
char operator;
float number1, number2, result;
printf(“Enter two numbers and an operator in the format\n”);
printf(“ number1 operator number2\n”);
scanf(“%f %c %f”, &number1, &operator, &number2);
if(operator == ‘*’)
result = number1 * number2;
else if(operator == ‘/’)
result = number1 / number2;
else if(operator == ‘+’)
result = number1 + number2;
else if(operator == ‘-‘)
result = number1 – number2;
else
invalid_operator = 1;
if( invalid_operator != 1 )
printf(“%f %c %f is %f\n”, number1, operator, number2, result );
else
printf(“Invalid operator.\n”);
}
```

Switch statement:

The switch case statement is a better way of writing a program when a series of if elses occurs.

The general format is,

```
switch ( expression ) {
case value1:
program statement;
program statement;
.....
```

```
break;
case valuen:
program statement;
.....
break;
default:
.....
.....
break;
}
```

The keyword break must be included at the end of each case statement. The default clause is optional, and is executed if the cases are not met. The right brace at the end signifies the end of the case selections.

Example:

```
#include <stdio.h>
main()
{
 int menu, numb1, numb2, total;
printf("enter in two numbers □");
scanf("%d %d", &numb1, &numb2 );
printf("enter in choice\n");
printf("1=addition\n");
printf("2=subtraction\n");
scanf("%d", &menu );
switch( menu )
{
case 1: total = numb1 + numb2; break;
case 2: total = numb1 – numb2; break;
default: printf("Invalid option selected\n");
}
 if( menu == 1
)
printf("%d plus %d is %d\n", numb1, numb2, total );
else if( menu == 2 )
printf("%d minus %d is %d\n", numb1, numb2, total );
}
```

The above program uses a switch statement to validate and select upon the users input choice, simulating a simple menu of choices.

10.Explain the following:

break statement with example program,
continue statement with example program,
goto statement with example program

break Statement

break statment is a jump statment in C and is generally used for breaking from a loop or breaking from a case as discussed in switch stament.

Break statement when encountered within a loop immediately terminates the loop by passing condition check and execution transfer to the first statment after the loop. In case of switch it terminates the flow of control from one case to another.

Also one important thing to remember about break is that it terminates only the innermost switch or loop construct in case of nested switch and loop variations.

The C source code below shows an simple application of break statement-

```
#include <stdio.h>

int main ()
{
 for (int i = 0; i<100; i++)
  {
    printf ("\n%d", i);
    if (i == 10); // This code prints value only upto 10.
    break;
  }

 return 0;
}
```

continue Statement

continue is a jump statement provided by C. It is analogus to break statement. Unlike break which breaks the loop, continue stament forces the next execution of loop bypassing any code in between.

For for staments it causes the conditional check and increment of loop variable, for while and do-while it passes the control to the conditon check jumping all the statements in between. Continue plays an important role in efficiently applying certain algorithms.

Below is a C source code showing the application of continue statement -

```
#include <stdio.h>

int main ()
{
 for (int i = 0; i<100; i++)
  {
    if (i == 10); // This code prints value only upto 9 even though loop executes 100 times.
     continue ;
    printf ("\n%d", i);
  }
return 0;
}
```

goto Statement

goto statement is a jump staments and is perhaps the most controversial feature in C. Goto in unstructured language such as basic was indispensable and the soul of the programming lanugage but in a structured language they are critcized beyond limit. The critics say that due to other jump statements in c goto is not all required and more over it destoys the structure of your program and make it unreadable. But on other hand goto statement of C provides some very powerfull programming options in very compicated situation. This tutorial will take an neutral stand in such an argument and neither recommend nor discourage use of goto.

Goto is used for jumping from one point to another point in your function. You can not jump from one function to another. Jump points or way points for goto are marked by label statements. Label statement can be anywhere in the function above or below the goto statement. Special situation in which goto find use is deeply nested loops or if - else ladders.

General form of goto statement is -

```
.
.
.
goto label1;
.
.
.
.
label1 :
.
.
.
label2 :
.
.
.
.
goto label2;
```

Examples:

```
goto skip_point;
printf("This part was skipped.\n");
skip_point:
printf("Hi there!\n");
```

UNIT – V
PART – A

1. What is meant by Recursive function?

If a function calls itself again and again, then that function is called Recursive function.

2. What is an array?

An array is a group of similar data types stored under a common name. int a[10];
Here a[10] is an array with 10 values.

3. What is a Pointer? How a variable is declared to the pointer?

Pointer is a variable which holds the address of another variable.

Pointer Declaration:

datatype *variable-name;

Example:

```
int *x, c=5;
x=&a;
```

4. What are the uses of Pointers?

Pointers are used to return more than one value to the function, Pointers are more efficient in handling the data in arrays, Pointers reduce the length and complexity of the program, They increase the execution speed, The pointers saves data storage space in memory

5. What is the output of the program?

```
main() junk(int i, int j)
{ {
int i=5;j=2; i=i*j;
junk(i,j); j=i*j;
printf("\n %d %d",i,j); }
}
```

Output:

1. 2
2.

6. What are * and & operators means?

'*' operator means 'value at the address'

'&' operator means 'address of'

7. What is meant by Preprocessor?

Preprocessor is the program, that process our source program before the compilation.

8. How can you return more than one value from a function?

A Function returns only one value. By using pointer we can return more than one value.

9. Is it possible to place a return statement anywhere in 'C' program?

Yes. The return statement can occur anywhere.

10. What are the main elements of an array declaration?

Array name, Type and Size

11. List the header files in 'C' language.

<stdio.h> contains standard I/O functions, **<ctype.h>** contains character handling functions, **<stdlib.h>** contains general utility functions, **<string.h>** contains string manipulation functions, **<math.h>** contains mathematical functions, **<time.h>** contains time manipulation functions

12. What are the steps involved in program development life cycle?

Program Design, Program Coding, Program Testing & Debugging

13. What are the types of errors occurred in C program?

Syntax errors, Runtime errors, Logical errors and Latent errors

14. What is testing?

Testing is the process of executing the program with sample or tested data.

15. What are the types of testing?

Human testing and Computer based testing

16. How do you define enumerated data type?

```
enum mar_status{ single,married,widow };
enum mar_status person1,person2;
person1=married;
```

Here the person1 is assigned to value zero.

17. What is meant by debugging?

Debugging is the process of locating and isolating the errors.

18. Specify any five syntax error messages.

Missing semicolon, Missing braces, Missing quotes, Improper comment characters and Undeclared variables

19. What are the pre-processor directives?

Macro Inclusion, Conditional Inclusion and File Inclusion

20. What is dynamic memory allocation?

Allocating the memory at run time is called as dynamic memory allocation.

21. What are the various dynamic memory allocation functions?

malloc() - Used to allocate blocks of memory in required size of bytes.

free() - Used to release previously allocated memory space.

calloc() - Used to allocate memory space for an array of elements.

realloac() - Used to modify the size of the previously allocated memory space.

22. What is the deference between declaring a variable and defining a variable?

Declaring a variable means describing its type to the compiler but not allocating any space for it. Defining a variable means declaring it and also allocating space to hold the variable. A variable Can also be initialized at the time it is defined. To put it simply, a declaration says to the compiler, "Some where in the program there will be a variable with this name, and this is the kind of data Type it is." On the other hand, a definition says, "Right here is this variable with this name and This data type". Note that a variable can be declared any number of times, but it must be defied Exactly once. For this reason, definitions do not belong in header files, where they might get #included into more than one place in a program.

23. Why does n++ execute than n=n+1?

The expression n++ requires a single machine instruction such as INR to carry out the increment operation whereas; n+1 requires more instructions to carry out this operation.

24. Why is it necessary to give the size of an array in an array declaration?

When an array is declared, the compiler allocates a base address and reserves enough space in the memory for all the elements of the array. The size is required to allocate the required space. Thus, the size must be mentioned.

25. Where in memory are variables stored?

Variables can be stored in several places in memory, depending on their lifetime.

(1) Variables that are defined outside any function (whether of global or file static scope), and variables that are defined inside a function as static variables, exist for the lifetime of the program's execution. These variables are stored in the data segment. The data segment is a fixed-size area in memory set aside for these variables.

(2) Variables that are the arguments functions exist only during the execution of that function. These variables are stored on the stack. The stack is an area of memory that starts out as small and grows automatically up to some predefined limit.

(3) The third area is the one that does not actually store variables but can be used to store data pointed to by variables. Pointer variables that are assigned to the result of a call to the function malloc() contain the address of a dynamically allocated area of memory. This memory is in an area called the heap.

26. What is an heap memory?

The heap is another area that starts out as small and grows, but it grows only when the programmer explicitly calls malloc() or other memory allocation functions, such as calloc(). The heap can share a memory segment with either the data segment or the stack, or it can have its own segment, it all depends on the compiler options and operating system. The heap, like the stack, has a limit on how much it can grow, and the same rules apply as to how that limit is determined.

27. What is the difference between an array and pointer?

Difference between arrays and pointers are as follows.

Array

Array allocates space automatically, It cannot be resized, It cannot be reassigned, Size of(array name) gives the number of bytes occupied by the array.

Pointer

Pointer is explicitly assigned to point to an allocated space, It can be resized using realloc (), Pointers can be reassigned, Sezeof(pointer name) returns the number of bytes used to store the pointer variable.

27. What is the purpose of the function main()?

The function main () invokes other functions within it. It is the first function to Be called when the program starts execution. Some salient points about main() are as follows: It is the starting function ,. It returns an int value to the environment that called the program, Recursive call is allowed for main() also, It is a user-defined function, Program exection ends when the closing brace of the function main() is reached, It has two arguments (a) argument count and (b)argument vector (reprensents strings passed.), Any user-defined name can also be used as parameters for main() instead of argc and argv

28. What is dangling pointer?

In C, a pointer may be used to hold the address of dynamically allocated Memory After this memory is freed with the free() function, the pointer itself will still contain the address of the released block. This is referred to as a dangling pointer. Using the pointer in this state is a serious programming error. Pointer should be assigned NULL after freeing

memory to avoid this bug.

29. Compare arrays and structures.

Comparison of arrays and structures is as follows.

Arrays : An array is a collection of data items of same data type.

Arrays can only be declared.

There is no keyword for arrays.

An array name represents the address of the starting element.

An array cannot have bit fields.

Structures : A structure is a collection of data items of different data types.

Structures can be declared and defined.

The Keyword for structures is struct.

A structrure name is known as tag. It is a shorthand notation of the declaration.

A structure may contain bit fields.

30. Compare structures and unions.

Structure

Every member has its own memory.

The keyword used is struct.

All members occupy separate memory location, hence different interpretations of the same memory location are not possible. Consumes more space compared to union.

Union

All members use the same memory.

The keyword used is union.

Different interpretations for the same memory location are possible.Conservation of memory is possible.

31. What is call by value? (JAN 2011)

Passing a variable by value makes a copy of the variable before passing it onto a function. This means that if you try to modify the value inside a function, it will only have the modified value inside that function. One the function returns, the variable you passed it will have the same value it had before you passed it into the function.

32. What is call by reference?

There are two instances where a variable is passed by reference:

1. When you modify the value of the passed variable locally and also the value of the variable in the calling function as well.

2. To avoid making a copy of the variable for efficiency reasons.

PART – B

1. **What are functions? Explain the types of functions in detail with an example program for each type.**

A function is a self contained block or a sub program of one or more statements that performs a special task when called.

Types:

- Pre defined functions.
- User Defined functions

User Defined functions

The functions defined by the user for their requirement are called user-defined functions.

Whenever it is needed, The user can modify the function.

Example: sum(a,b)

Advantage of User-Defined Functions

- The length of the source program can be reduced.
- It is easy to locate error.
- It avoid coding of repeated instructions.

Elements of User-Defined Function

(a) Function Declaration
returntype function-name(Parameters);
Example:
int square(int, int);
(b) Function calling
function-name(actual parameters);
Example:
int square(a,b);
(c) Function Definition:
returntype function-name(formal parameters)
{
local variable declaration;
statement 1;
statement 2;
return(value);
}
Example:
void square(int a, int b)
{
printf("%d",(a*b));
}

Pre defined functions.
Also called library function.
Examples:
sqrt(x):
It is used to find the square root of **x**
Example: **sqrt(36)** is 6
abs(x):
It is used to find the absolute value of **x**
Example: **abs(-36)** is 36
pow(x,y):
It is used to find the value of $\mathbf{x^y}$
Example: **pow(5,2)** is 25
ceil(x):
It is used to find the smallest integer greater than or equal to **x**
Example: **ceil(7.7)** is 8

2. Write a C program to check whether the given number is palindrome or not. (JAN 2012)

```
#include<stdio.h>
#include<math.h>
void main()
{
    long int n, num, rev = 0, dig;
    clrscr();
    printf("\n\n\t ENTER A NUMBER...: ");
    scanf("%ld", &num);
    n = num;
    while(num>0)
    {
        dig = num % 10;
        rev = rev * 10 + dig;
        num = num / 10;
```

```
    }
    if (n == rev)
        printf("\n\t GIVEN NUMBER IS A PALINDROME");
    else
        printf("\n\t GIVEN NUMBER NOT A PALINDROME");
    getch();
}
```

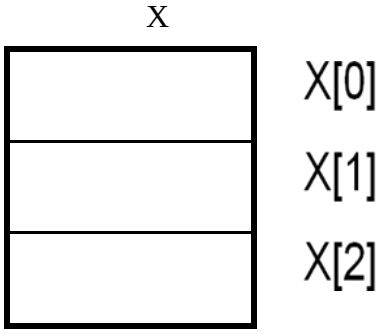
3. Define arrays. Explain the array types with an example program for each type.
   An Array is a collection of similar data items, that are stored under a common name.

   Types

   - One-Dimensional array
   - Two-Dimensional array
   - Multi-Dimensional array

   Syntax:

   data_type array_name[size];
   Example: int x[3];



Array initialization

At compile time

At run time

At compile time

Syntax:

data_type array_name[size]={variables};

Example: int x[3]={5,3,7};

At Run time

Array can also initialize at the run time.

Example

```
#include<stdio.h>

#include<conio.h>

void main()
```

```
{
 int x[2],i;
 printf("\nEnter the inputs:");
 for(i=0;i<2;i++)
        scanf("%d",&x[i]);
 for(i=0;i<2;i++)
        printf("\nThe value in x[%d] is %d",i,x[i]);
 getch();
}
```

Two-Dimensional array

Array Declaration

Syntax:

data_type array_name[row_size] [col_size];

Example: int x[3][2];

Array Initialization

Syntax:

data_type array_name[row_size] [col_size];={variables};

Example: int x[2][2]={1,50,2,75};

Example

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int i,j;
 int x[2][2]={  {1,50},
                {2,75}
             };
 clrscr();
 for(i=0;i<2;i++)
        for(j=0;j<2;j++)
        printf("\nThe value in x[%d][%d] is %d",i,j,x[i][j]);
 getch();
}
```

4.Explain the standard string functions with example to support each type.

strlen()

It is used to find the length of the string.

syntax:

strlen(string);

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
 char a[]="college";
 int b;
```

```
 clrscr();
 b=strlen(a);
 printf("\nThe length of the string is  %d",b);
 getch();
}
```

Output:

The length of the string is 7

strcpy()

It is used to copy one string to another.

syntax:

strcpy(string1,string2) ;

Example:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
 char a[]="IT";
 char b[]="Dept";
 clrscr();
 strcpy(a,b);
 printf("\nThe string is  %s",a);
 getch();
}
```

Output:

The string is Dept

strcat()

It is used to combine two strings.

syntax:

strcat(string1,string2) ;

Example:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
 char a[]="IT";
 char b[]="Dept";
 clrscr();
 strcat(a,b);
 printf("\nThe string is  %s",a);
 getch();
}
```

Output:

The string is ITDept

strcmp()

It is used to compare two strings.

syntax:

strcmp(string1,string2)

- Returns 0 if two strings are equal.
- Return value <0 if s1 is less than s2.
- Return value >0 if s1 is greater than s2.

Example:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
char a[]="itdept";
char b[]="it";
int i;
clrscr();
i=strcmp(a,b);
if(i==0)
        printf("\nstrings are equal:%d",i);
else if(i<0)
        printf("\nstring1 is less than string2:%d",i);
else
        printf("\nstring1 is greater than string2:%d",i);
 getch();
}
```

Output:

string1 is greater than string2:100

strlwr(), strupr()

It used to change the case of a string.

syntax:

```
strlwr(string);
strupr(string);
```

Output:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
char a[]="itdept";
clrscr();
printf("\nThe string is :%s",a);
strupr(a);
printf("\nThe string after conversion to uppercase :%s",a);
strlwr(a);
printf("\nThe string after conversion to lowercase :%s",a);
 getch();
}
```

Output:

The string is :itdept
The string after conversion to uppercase :ITDEPT
The string after conversion to lowercase :itdept

strrev()

It used to reverse a string.

syntax:

```
strrev(string)
```

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
 char a[]="Dept";
clrscr();
printf("\nThe string is  %s",strrev(a));
 getch();
}
```

Output:

The string is tpeD

5. What are pointers? When and why they are used? Explain in detail with sample programs. (JAN 2011)

Pointer is a variable that contains the memory address of another variable.

Eg: int *a;

Uses

1.Dynamic memory allocation is possible with pointers.

2.passing arrays and structures to functions

3.passing addresses to functions.

Accessing Variable through Pointer

Example:

```
int *a;
x=5;
a=&x;
```

Exmple program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int x=5;
 int *a;
 a=&x;
 printf("\n The Value of x = %d",x);
 printf("\n The Address of x = %u",&x);
 printf("\n The Value of a = %d",a);
 printf("\n The Value of x = %d",*a);
}
```

Null Pointer

A pointer is said to be null pointer if zero is assigned to the pointer.

Example

```
int *a,*b;
a=b=0;
```

Pointer to Pointer

Here one pointer stores the address of another pointer variable.

Example:

```
int x=10,*a,**b;
a=&x;
b=&a;
```

Pointer to Array

The elements of the array can also be accessed through a pointer

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int a[3]={2,3,7};
 int *b;
 b=a;
 printf("\n The Value of a[0] = %d",a[0]);
 printf("\n The Address of a[0] = %u",&a[0]);
 printf("\n The Value of b = %d",b);
}
```

Pointer to Structures

Syntax:

```
struct structure_name
{
        structure element1;
        structure element2;
        ……….
}variable,*ptr;
```

6. Describe in detail about the Preprocessors in C.

Preprocessor

It is a program that processes the source program before compilation.

It operates under the following directives

File Inclusion
Macro substitution
Conditional inclusion

File Inclusion

It is used to include some file that contains functions or some definitions.

Syntax:

```
#include<filename>   (or)
#include“filename”
```

Eg:

```
 #include<stdio.h>
 #include “ex.c”
```

Example:

```
#include<stdio.h>
#include<conio.h>
#include "addition.txt"
void main()
{
 int a,b;
 printf("\nEnter the numbers:");
 scanf("%d%d",&a,&b);
 printf("The Value is %d",add(a,b));
 getch();
}
```

addition.txt

```
int add(int a,int b)
```

```
{
 return(a+b);
}
```

Macro Substitution

It is used to define and use integer, string, or identifier in the source program

The three forms of macros are

- Simple Macro
- Argumented Macro
- Nested Macro

Simple Macro

It is used to define some constants

Syntax

```
# define identifier string/integer
```

Eg:

```
#define pi 3.14
#define CITY “chennai”
```

Example program:

```
#include<stdio.h>
#include<conio.h>
#define pi 3.14
#define CITY "chennai"
void main()
{
 printf("The Value is %f",2*pi);
 printf("\nThe Value CITY is %s",CITY);
 getch();
}
```

Output:

```
The Value is 6.280000
The Value CITY is chennai
```

Argumented Macro

It is used to define some complex forms in the source program.

Syntax:

```
#define identifier (v1,v2,….) string/integer
```

Eg:

```
#define cube(n) (n*n*n)
```

Example:

```
#include<stdio.h>
#include<conio.h>
#define cube(n) (n*n*n)
void main()
{
 printf("The Value of 3 cube is %d",cube(3));
 getch();
}
```

Output:

```
The Value of 3 cube is 27
```

Nested Macro

Here one macro is used by another macro.

Eg:

```
#define a 3
#define sq a*a
```

Conditional Inclusion

It is used to include some conditional statements**.**

Example:

```
#include<stdio.h>
#include<conio.h>
#define a 3
#ifdef a
#define c a+5
#endif
void main()
{
 printf("\nThe value C is %d",c);
 getch();
}
```

Output:

The value C is 8

7. Brief call by value and call by reference in detail.

Pointers and Functions

- Call by Value
- Call by Reference

Call by value

Actual argument passed to the formal argument.

Any changes to the formal argument does not affect the actual argument.

```
#include <stdio.h>
#include<conio.h>
void main()
{
 int x,y,swap(int,int);
 printf("\nEnter value of x:");
 scanf("%d",&x);
 printf("\nEnter value of y:");
 scanf("%d",&y);
 change(x,y);
 printf("\n\nValues in the Main()-->x=%d,y=%d",x,y);
 }
 int swap(int a,int b)
{
 int c;
 c=a;
 a=b;
 b=c;
 printf("\nValues in the Function -->x=%d,y=%d",a,b);
}
```

Output

```
Enter value of x:5
Enter value of y:6
```

Values in the Function -->x=6,y=5
Values in the Main()-->x=5,y=6

Call by reference

Instead of passing value, the address of the argument will be passed.
Any changes to the formal argument will affect the actual argument.

Example

```
#include <stdio.h>
#include<conio.h>
void main()
{
        int x,y,change(int*,int*);
        printf("\nEnter value of x:");
        scanf("%d",&x);
        printf("\nEnter value of y:");
        scanf("%d",&y);
        change(&x,&y);
        printf("\n\nValues in the Main()-->x=%d,y=%d",x,y);
}
int change(int *a,int *b)
{
        int c;
        c=*a;
        *a=*b;
        *b=c;
 printf("\nValues in the Function -->x=%d,y=%d",*a,*b);
}
```

Output:

```
Enter value of x:5
Enter value of y:6
Values in the Function -->x=6,y=5
Values in the Main()-->x=6,y=5
```

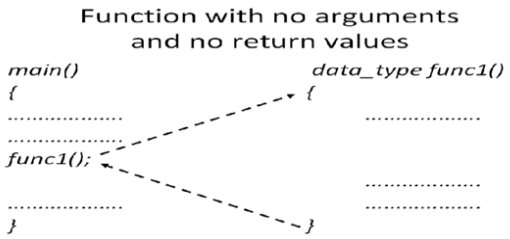
8. Discuss about function prototypes in detail

Function Prototypes

- Function with no arguments and no return values.
- Function with arguments and no return values.
- Function with arguments and return values.
- Function with no arguments and with return values.

Function with no arguments and no return values

- Here no data transfer take place between the calling function and the called function.
- These functions act independently, i.e. they get input and display output in the same block.



Example:

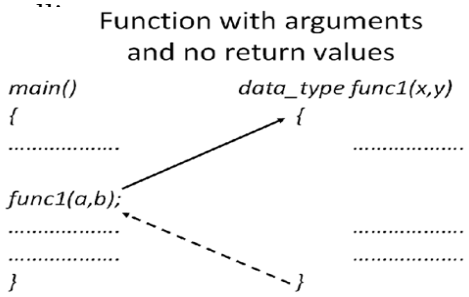
```
#include <stdio.h>
#include<conio.h>
void main()     //calling function
{
        void add(void);
        add();
}
void add()      //called function
{
inta,b,c;
printf("\nEnter two number:");
scanf("%d%d",&a,&b);
 c=a+b;
printf("\nSum is:%d",c);
}
```

Output:

```
Enter two number:3
4
Sum is:7
```

Function with arguments and no return values

- Here data transfer take place between the calling function and the called function.
- It is a one way data communication, i.e. the called program receives data from calling program but it does not return any value to th



Example:

```
#include <stdio.h>
#include<conio.h>
void main()
{
inta,b;
void add(int,int);
printf("\nEnter two number:");
scanf("%d%d",&a,&b);
```

```
add(a,b);
}
void add(intx,int y)  //function with arguments
{
int z;
 z=x+y;
printf("\nSum is:%d",z);
}
```

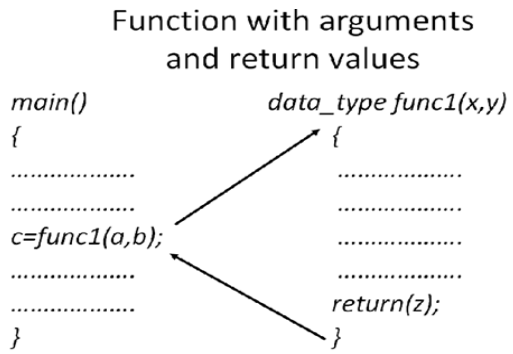
Output:

```
Enter two number:2
4
Sum is:6
```

Function with arguments and return values

Here data transfer take place between the calling function and the called function as well as between called function and calling function .

It is a two way data communication, i.e. the called program receives data from calling program and it return some value to the calling program.



Example:

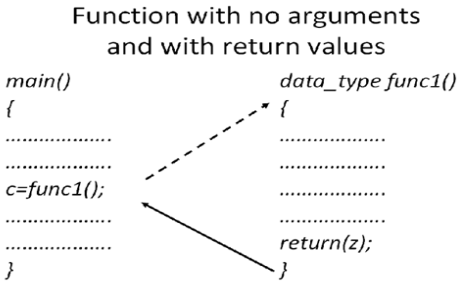
```
#include <stdio.h>
#include<conio.h>
void main()
{
inta,b,c;
int add(int,int);
printf("\nEnter two number:");
scanf("%d%d",&a,&b);
c=add(a,b);
printf("\nSum is:%d",c);
}
intadd(intx,int y)
{
int z;
z=x+y;
return(z);
}
```

Output:

```
Enter two number:6
7
Sum is:13
```

Function with no arguments and with return values

- Here data transfer take place between the called function and the calling function.
- It is a one way data communication, i.e. the called program does not receives data from calling program but it return some value to the calling program.



Example:

```
#include <stdio.h>
#include<conio.h>
void main()
{
int add(),d;
d=add();
printf("\nSum is:%d",d);
}
int add() //function wit no argument
{
inta,b,c;
printf("\nEnter two number:");
scanf("%d%d",&a,&b);
c=a+b;
return(c);
}
```

Output:

```
Enter two number:5
8
Sum is:13
```

9. Write a C program to generate Fibonacci series upto 200. (JAN 2012)

```
#include<stdio.h>
 #include<conio.h>
void main()
 {
 long int a,b,c,i;
 clrscr();
 a=-1;
 b=1;
 printf("\n FIBONACCI SERIES .....\t");
 i=1;
 while(i<=200)
 {
 c=a+b;
 printf("\n%ld",c);
```

```
a=b;
b=c;
i++;
}
getch();
}
```

10. Define Structures. Explain structures in detail. (JAN 2011)

A **Structure** is a collection of different data items, that are stored under a common name.

Syntax:

```
struct structure_name
{
        structure element1;
        structure element2;
        ……..
};
```

Example:

```
struct stud
{
        int sno;
        char name[10];
        int mark;
};
struct stud s;
```

```
#include<stdio.h>
#include<conio.h>
struct stud
        {
                int regno;
                char name[10];
                int m1;
                int m2;
                int m3;
        };
struct stud s;
void main()
{
 float tot,avg;
printf("\nEnter the student regno,name,m1,m2,m3:");
scanf("%d%s%d%d%d",&s.regno,&s.name,&s.m1,&s.m2,&s.m3);
 tot=s.m1+s.m2+s.m3;
 avg=tot/3;
 printf("\nThe student Details are:");
 printf("\n%d\t%s\t%f\t%f",s.regno,s.name,tot,avg);
 }
```

Structure assignment

It is possible to assign one structure information to another structure of same type using simple assignment statement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
```

```
        struct
        {
         int a;
        }x,y;
        clrscr();
        x.a=10;
        y=x;
        printf("The value of y.a is%d",y.a);
        getch();
 }
```

Output:

The value of y.a is10

11. Define Union. Explain Union in detail. (JAN 2011)

An **Union** is a collection of different data items, that are stored under a common name. Here same memory is shared by its members.

Syntax:

```
union union _name
{
        union element1;
        union element2;
        ……
};
```

Example:

```
 union result
{
        int mark;
        float avg;
        char grade;

};
union result s;
```

Example program:

```
#include<stdio.h>
#include<conio.h>
union stud
    {
        int a;
        char b[2];
    };
void main()
{
    union stud c;

c.a=256;
    printf("\nc.a value is%d",c.a);
    printf("\nc.b[0] value is%d",c.b[0]);
    printf("\nc.b[1] value is%d",c.b[1]);
 }
```

Output:

```
c.a value is256
c.b[0] value is0
c.b[1] value is1
```

12. Write C program to print the prime numbers less than 500. (JAN 2012)

```
#include<stdio.h>
#include<conio.h>
void main()
{
int count,i=1;
int a;
clrscr();
while(i<=500)
{
count=0;
a=1;
while(a<=i)
{
if(i%a==0)
count++;
a++;
}
if(count==2)
printf("%d\t",i);
i++;
}
getch();
}
```

13. Write a C program to find the factorial of a given number using recursion. (JAN 2012)

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int a;
 int rec(int);
 printf("\nEnter the number:");
 scanf("%d",&a);
 printf("The factorial of %d! is %d",a,rec(a));
}
int rec(int x)
{
 int f;
 if(x==1)
     return(1);
 else
     f=x*rec(x-1);
 return(f);
}
```

Output:

Enter the number:5
The factorial of 5! is 120